UNIVERSIDADE FEDERAL DE SANTA CATARINA

SysLODFlow - Ampliando as funcionalidades na automação da publicação e manutenção de dados abertos conectados usando o LodFlow

Bruno Eduardo D'Angelo de Oliveira

Florianópolis - SC
2017/2

UNIVERSIDADE FEDERAL DE SANTA CATARINA

DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

SysLODFlow: Ampliando as funcionalidades na automação da publicação e manutenção de dados abertos conectados usando o LodFlow

Bruno Eduardo D'Angelo de Oliveira

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal de Santa Catarina.

Orientador: Prof. Dr. José Leomar Todesco

Florianópolis - SC
2017/2

Bruno Eduardo D’Angelo de Oliveira

SysLodFlow: Ampliando as funcionalidades na automação da publicação e manutenção de dados abertos conectados usando o LodFlow

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação pela Universidade Federal de Santa Catarina.

Orientador: Prof. Dr. José Leomar Todesco

Banca examinadora:

Prof. Dr. José Leomar Todesco,
Orientador
Universidade Federal de Santa Catarina

Prof. Dr. Fernando Alvaro Ostuni Gauthier,
Universidade Federal de Santa Catarina

Ma. Lidiane Visintin,
Universidade Federal de Santa Catarina

AGRADECIMENTOS

Ao meu orientador pelos ensinamentos, paciência e por ter se mantido confiante em minha capacidade até o final.

A Juana Goulart Stollmaier pois sem seu olhar crítico, disposição e amizade este trabalho nunca teria sido concluído de maneira satisfatória.

A minha namorada Eduarda de Moraes que sempre se manteve a meu lado e que sua ternura e carinho foram essenciais para me manter focado e disposto a enfrentar os obstáculos que surgiram em meu caminho.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SÍMBOLOS

CSV - *Comma Separated Values*
BD - Banco de Dados
RDF - *Resource Description Framework*
LDWMS - *Linked Data Workflow Management System*
ES - Engenharia de Software
UX - *User Experience*
JSON - *JavaScript Object Notation*
W3C - *World Wide Web Consortium*
XML - *Extensible Markup Language*
URI - *Uniform Resource Identifier*
HTTP - *Hypertext Transfer Protocol*
LOD - *Linked Open Data*
SPARQL - *SPARQL Protocol and RDF Query Language*
LDWPO - *Linked Data Worflow Project Ontology*
SO - Sistema Operacional
CSS - *Cascade Style Sheet*
XHTML - *Extensible Hypertext Markup Language*
TCC - Trabalho de Conclusão de Curso
IDE - *Integrated Development Environment*
GUI - *Graphical User Interface*
JEE - Java *Enterprise Edition*
JAR - Java *Archive*
OWL - *Web Ontology Language*
API - *Application Programming Interface*

RESUMO

O conceito de dados conectados surgiu para que a publicação e difusão de informações na *web* pudessem ser aprimoradas. Utilizando estes conceitos é possível ligar dados relevantes, fazendo com que seja possível explorar as relações entre essas informações. Diversas ferramentas foram desenvolvidas com o intuito de auxiliar na extração e limpeza de dados, contudo todas exigem conhecimentos específicos sobre sua aplicação, e a necessidade de integração com outras soluções de terceiros. Dessa maneira o processo para a divulgação do conhecimento gerado seguindo os preceitos de *Linked Data* se torna extremamente oneroso. Para tentar abrandar a dificuldade encontrada no processo foi criada a ferramenta *SysLODFlow*, a qual integra diversas soluções da área, criando um fluxo de trabalho que engloba todas as etapas do processo de publicação. Entretanto o *software* desenvolvido apresenta limitações em suas funcionalidades, as quais prejudicam a experiência de utilização. O objetivo deste trabalho é rever a concepção da ferramenta, analisar as necessidades pendentes, e criar as devidas soluções aos problemas averiguados.

Palavras-chave: *Linked Data*, Dados Conectados, Ontologias, *LODFlow*, *SysLODFlow*, Engenharia de *Software*, Engenharia Reversa.

1. INTRODUÇÃO

O conceito de *Linked Data* (dados conectados) é um conjunto de práticas introduzidas por Tim Berners-Lee (2006), que tem como objetivo utilizar a *web* para relacionar dados que previamente não tinham ligação, ou usar os recursos *web* para ao menos diminuir as barreiras entre as mesmas. Com isso em mente pode-se aplicar as técnicas descritas a grandes massas de dados, como, por exemplo, os disponibilizados pelo governo federal Brasileiro, assim possibilitando uma exploração mais fluida, e a possível descoberta, ou criação, de conhecimento a partir dos resultados obtidos.

No entanto, junto com todas as vantagens que o seguinte modelo proposto fornece, surgem as dificuldades em se realizar a manutenção dos mesmos, como evidenciado em diversos estudos sobre o assunto de *Linked Data* (AUER et al., 2013). Sendo essa uma atividade complicada e que envolve diversos passos que demandam ferramentas específicas. Porém o custo dessas operações pode vir a ser reduzido com a introdução da automatização do fluxo de trabalho e publicação. Para se atingir esse objetivo foi desenvolvido o *software* LODFlow (RAUTENBERG et al. 2015), o qual proporciona métodos para suportar o ciclo de vida destes conjuntos de dados de uma forma sistemática. Mas mesmo utilizando essa solução a carga imposta aos usuários ainda era considerada elevada, pois o tempo necessário para compreensão de seu uso era alto, e ainda assim era necessário utilizar outras aplicações.

Com o intuito de automatizar ainda mais o fluxo de trabalho criou-se o *SysLODFlow*, que faz uso das funcionalidades do LODFlow, mas fornece opções mais "*user friendly*" (e automatizadas) para se realizar o ciclo de publicação e manutenção de dados conectados. A ferramenta também foi desenvolvida de maneira a ser um aplicativo web, o que soluciona algumas restrições iniciais do LODFlow, e abrem as portas para funcionalidades que antes não eram possíveis em aplicações *standalone*, mas que trazem consigo algumas necessidades pontuais de design para web.

Todavia existem limitações no escopo de operação da aplicação que precisam ser tratadas e desenvolvidas, para que o programa possa melhor suprir as necessidades de seu público alvo. Dessa forma este trabalho pretende estudar a implementação do SysLODFlow, entender suas escolhas de design, propor e desenvolver, sempre se utilizando de boas práticas de engenharia e programação de *software*, soluções para as deficiências encontradas na iteração atual do projeto, bem como elaborar um documento que explique o funcionamento básico das funcionalidades do projeto.

1.1. Problema

Como já mencionado, o esforço demandado para se realizar trabalhos de *Linked Data*, agregado com o grande número, complexidade, e baixa intuitividade de uso das ferramentas disponíveis para a área, faz com que a barreira para publicação de dados na *web*, seguindo as práticas descritas por Berners-Lee (2006), seja de difícil transposição. Com o intuito de abrandar a carga de trabalho, e conhecimento, imposta ao usuário final, criou-se a aplicação SysLODFlow (MORAIS; MORAIS, 2016). Sendo esta uma solução que agrega (de forma a criar um fluxo de trabalho que integra as diferentes partes que a constituem) diversos outros *softwares,* os quais são aplicados durante todo o ciclo de publicação de dados conectados, para assim diminuir o encargo teórico e prático que recai sobre o utilizador.

Contudo, o programa desenvolvido é, em diversas áreas, limitado em seu funcionamento. Exemplificando o estado incompleto da ferramenta temos como grande problema o fato de apenas arquivos em formato CSV serem aceitos, como entrada para processamento. Sendo assim volumes verdadeiramente significativos, como aqueles encontrados em bancos de dados, não são de fáceis de se tratar utilizado o SysLODFlow, e formatos mais recentes e práticos, como o JSON, necessitam de conversão para o CSV, o que aumenta a carga de trabalho para o usuário, o que acaba sendo contra-produtivo.

Ademais, temos outros problemas notáveis: funcionalidades necessárias não são suportadas, documentação básica de instalação e uso é inexistente, e é necessário se criar interfaces para geração e edição dos diversos arquivos de configuração dos *softwares* integrados ao fluxo de trabalho.

1.2. Motivação

O SysLODFlow foi concebido com a visão de ser um artefato de *software* que possibilita-se realizar o processo de maneira mais integrada e intuitiva para os usuários finais. Porém a sua operação ainda é limitada e, por vezes, necessita que se opere diretamente com as configurações das ferramentas de apoio, fato que compromete a missão inicial do projeto. A partir do estudo do código fonte, funcionamento do programa e da leitura do relatório de projeto gerado, foram levantadas melhorias necessárias para que o aproximar o SysLodFlow de sua proposta original.

1.3. Justificativa

Consequentemente este trabalho visa detalhar o funcionamento do *software* original em sua elaboração, explorar as limitações e aspectos a serem aprimorados, que foram encontrados, e propor melhorias para esses. Por fim espera decidir-se pontos a serem executadas e desenvolvidos, com intuito de corrigir os itens mais críticos ao bom funcionamento da aplicação e evoluir funcionalidades chave que já estejam em operação.

1.4. Objetivos

Este trabalho tem como objetivo o estudo e aprimoramento de uma ferramenta de publicação de dados conectados.

1.4.1. Objetivo Geral

Desenvolver aperfeiçoamentos necessários para uma melhor, e mais completa, experiência de uso da aplicação SysLODFlow por parte dos usuários finais.

1.4.2. Objetivos Específicos

- Desenvolver uma visão geral do estado da aplicação, através da engenharia reversa do código;
- Elaborar melhorias no fluxo de operação da ferramenta e/ou, criar novas funcionalidades que melhorem o uso da mesma;
- Corrigir erros críticos encontrados, que possam vir a fazer com que a experiência de uso da aplicação seja melhor e mais segura;
- Assegurar-se que quaisqueres interfaces gráficas produzidas sejam condizentes com a estética geral da aplicação, mas que façam uso de boas práticas de *User Experience* sempre que possível;
- Criar documentação inicial, que detalhe, ao menos, o processo de instalação, e como as funcionalidades básicas são operadas;
- Submeter o código desenvolvido para ser integrado ao projeto original, disponibilizado pelos idealizadores do software (MORAIS; MORAIS, 2016), no repositório da plataforma GitHub[1].

[1] GitHub é uma plataforma de hospedagem de código para controle de versão e colaboração, baseado no sistema de versionamento Git (github.com).

1.5. Método de Pesquisa

- Coletar e analisar os assuntos críticos para o desenvolvimento do trabalho;
- Estudar o código fonte da aplicação mencionada;
- Elaborar e implementar melhorias para a ferramenta;
- Documentar as aperfeiçoamentos realizados durante o andamento do projeto.

1.6. Resultados Esperados

Após a conclusão deste trabalho, espera-se ter alcançado uma melhoria substancial na qualidade, e variedade, das funcionalidades disponibilizadas pelo SysLODFlow.

1.7. Organização do Texto

Este documento está dividido em 5 capítulos, sendo este primeiro capítulo sobre a apresentação do projeto, no qual se detalha o problema vislumbrado, as motivações, justificativa e outros procedimentos metodológicos relevantes para este projeto.

No segundo capítulo é apresentada a fundamentação teórica do trabalho, que visa apresentar o estado da arte das áreas relacionadas ao tema. São apresentados assuntos como Engenharia de Software, *Web Semântica* e as ferramentas LODFlow e SysLODFlow - as quais são o tema central desta obra.

A terceira seção destina-se ao diagnóstico da ferramenta SysLODFlow, tentando expor, da melhor maneira possível, o estado atual do aplicativo. Ademais discutem-se quais os pontos falhos do programa e quais melhorias podem ser realizadas em suas funcionalidades e quais podem ser criadas do zero.

A próxima parte deste documento detalha a proposta de desenvolvimento da ferramenta SysLODFlow. São abordadas as soluções levantadas no segmento anterior que decidiu-se implementar, bem como os motivos para tal. Além disso, o processo de execução das tarefas é documentado e discutido.

Por fim o último capítulo trata dos resultados do desenvolvimento das melhorias propostas, dos trabalhos futuros que podem ser realizados em cima do que foi realizado, e apresenta as considerações finais do projeto.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo descrever o estado da arte relacionado a engenharia de software, e da publicação de dados abertos conectados.

A princípio serão abordados os temas relacionados a engenharia de *software*, como sua importância e as técnicas envolvidas no processo.

Finalizando essa parte, o próximo passo é discutir a Web Semântica, pois é uma ponto fortemente relacionado aos conceitos chave deste trabalho, e é necessário que se mantenham seus fundamentos em mente ao se desenvolver as atividades propostas neste documento.

Após cobrir-se esse assunto, serão discutidos os aspectos de *Linked Data*, passando pelas diversas questões importantes que sejam relacionadas a este trabalho, como Ontologias e o próprio conceito de dados abertos.

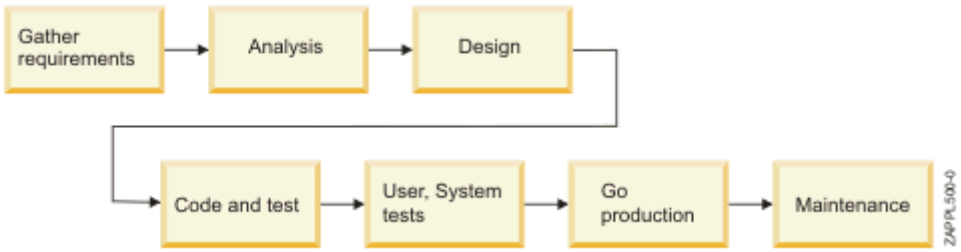
Finalmente serão discutidas as ferramentas LODFlow e SysLODFlow, para que se entendam tanto as motivações, quanto os propósitos que levaram à sua concepção e, também, suas arquiteturas.

2.1. Engenharia de Software

Conforme o tamanho e complexidade das aplicações aumentam com o passar dos anos, os problemas enfrentados em seu processo de criação deixam de ser relacionados a seus algoritmos ou estrutura de dados, passando a ser relacionados ao modo como o software como um todo foi arquitetado. Surge então a necessidade de um planejamento meticuloso da estrutura do projeto, que leve em conta desenvolvimento futuro e presente. Portanto, diversos estudos foram realizados para se chegar a uma disciplina que ajudasse a guiar os desenvolvedores durante o processo de design, nascia assim o campo da Engenharia de software.

Engenharia de software é um termo cunhado por Bauer (1988), que define a área como “[...] a criação e a utilização de sólidos princípios de engenharia a fim de obter software de maneira econômica, que seja confiável e que trabalhe em máquinas reais". Em sua concepção os pilares básicos da disciplina englobam, segundo Boehm (1983), temas como: desenvolvimento iterativo, a realização de validações constantes, o uso de práticas modernas de programação e a manutenção do comprometimento com o processo como um todo. Dessa maneira, o termo abrange todos os aspectos do ciclo clássico de desenvolvimento de software, como a criação do projeto inicial, análise, desenvolvimento e manutenção. Na figura 1 podemos observar o ciclo em sua totalidade e a progressão de suas fases.

Figura 1: Ciclo de desenvolvimento de aplicações.



Fonte: IBM (2006).

Contudo com a passagem do tempo o campo se expandiu e amadureceu, o que possibilitou o surgimento de diversas técnicas e padrões que elevaram o patamar da qualidade do software disponível no mercado. A área também passou por um deslocamento de seu foco, de programas *standalone*, os quais se tornam obsoletos, para aplicativos *web*, que tem prioridades distintas e muitas vezes mais estritas do que era observado anteriormente.

A transformação descrita se dá pela prevalência da necessidade de se conectar diversos usuários aos sistemas, a popularidade de plataformas *mobile*, necessidade de acesso das funcionalidades a qualquer momento, e em qualquer lugar, e a ascensão dos *e-commerces* e *e-business*. Para estes casos de desenvolvimento existem peculiaridades específicas na área de ES, que não estão presentes no processo clássico de criação de software. Em especial as técnicas voltadas para UX e segurança de dados tomam maior destaque durante o processo de criação, e na área acadêmica, por sua crescente importância na tentativa de se criar uma web mais acessível e segura, para todos os tipos diferentes de usuários.

Dessa maneira a Engenharia de Software toma destaque maior no cenário global, por sua crescente influência em um mundo cada vez mais conectado à internet. A qual, por sua vez, se torna um lugar mais saturado de todos os tipos de conteúdo disponível aos seus usuários, sejam elas aplicativos ou mídias como imagens e/ou vídeos.

2.2. Engenharia Reversa de *Software*

Em seu sentido mais amplo, engenharia reversa se refere ao processo de extração de conhecimento, ou design, de qualquer artigo construído por humanos para que seja possível reproduzir o objeto, ou criar algo baseado no que foi aprendido.

No mundo da tecnologia, o termo “engenharia reversa” tem suas origens na análise de *hardware*, onde a necessidade de se extrair configurações de peças terminadas é corriqueira. Os atores van Deursen e Burd (2005) descrevem a área se preocupando com três questões chave:

1. Quais são representações de alto nível adequadas ao sistema de *software*?
2. Como uma determinada representação de alto nível pode ser obtida de forma sistemática a partir de fontes de um sistema ou traços de execução?
3. Como a representação obtida pode ser apresentada aos engenheiros de software para ajudá-los no trabalho de desenvolvimento de software do dia-a-dia?

Ao ajudar a responder essas perguntas, as tecnologias e métodos de engenharia reversa acabam por auxiliar em contextos da engenharia de *software* convencional, como: a compreensão de programas, migração de sistemas e evolução de *software*.

Para esclarecer melhor as utilidade do tema, alguns pesquisadores (CIPRESSO; STAMP, 2010) citam certos cenários que exigem a aplicação de engenharia reversa, como, por exemplo, um módulo antigo com diversas regras de negócio implementadas, que precisa ser atualizado (ou consertado), mas sua documentação é defasada e/ou inexistente, deixando como única fonte de informação o código fonte. Outros possíveis contextos para o uso dessas técnicas seriam na detecção e neutralização de programas maliciosos (vírus e *malwares*) e, de maneira similar, quando se deseja testar as medidas de segurança de um produto finalizado.

Ferramentas de assistência são cruciais no processo como um todo, já que por vezes é necessário lidar com código binário, ou legado, contudo os instrumentos à disposição no setor são rudimentares e proveem o mínimo necessário. Assim sendo, a noção de necessidade de profissionais capacitados é reforçada, ainda mais quando levamos em conta a realidade de que as ferramentas para auxiliar na execução das tarefas de engenharia reversa contam com o entendimento e destreza do engenheiro de *software* que as opera, ou nas palavras de outros autores:

> Even though several tools already exist to aid software engineers with the program understanding process, the tools focus on transferring information about a software system's design into the mind of the developer. The expectation is that the developer has enough skill to efficiently integrate the information into his/her own mental model of the system's architecture. (CIPRESSO; STAMP, 2010).

Dessa maneira, podemos observar a necessidade da existência de profissionais com essas competências, caso se deseje lidar de maneira adequada com as situações descritas.

2.3. *Web* Semântica

Como discutido anteriormente, a *web* tem desempenhado um papel cada vez maior na vida moderna, e com o aumento do tráfego nesse meio é natural que se eleve, também, a quantidade de informação disponível na rede. Tal evento se deve ao fato de que utilizamos o ambiente virtual como um repositório para diversos tipos de arquivos e dados digitais. No entanto, apesar de termos disponível esse grande mar de informações, os documentos em si não agregam valor semântico em seus conteúdos, o que torna a utilização desses dados uma tarefa árdua, e a sua pesquisa e referenciamento (de maneira precisa) quase impossível.

Quando a *internet* foi desenvolvida, seu propósito era criar não apenas um meio para comunicação humana, mas de máquinas também. Enquanto a rede realmente se tornou um dos principais meios de comunicação humana, a comunicação entre máquinas não teve o mesmo desenvolvimento, criando dificuldades em áreas como a busca de documentos. Com intuito de enfrentar esse crescente problema foi publicado o conceito de *Web Semântica*, por Tim Berners-Lee, James Hendler e Ora Lassila (2001). A proposta deste novo paradigma para a *web* é a de que, para conseguirmos encontrar as informações que desejamos na vastidão da internet, necessitamos de uma maneira mais eficiente de publicação das informações em formato digital, que permita que autômatos nos auxiliem no processo de descobrimento de conhecimento. Sendo assim um modo de atingirmos isso seria atribuir um significado aos dados publicados na rede, de maneira que seja perceptível tanto a humanos quanto para máquinas. Para compreendermos melhor a noção proposta, também podemos observar a forma com que a W3C Brasil descreve a *Web Semântica*:

> A Web Semântica dá às pessoas a capacidade de criarem repositórios de dados na Web, construírem vocabulários e escreverem regras para interoperarem com esses dados. A linkagem de dados é possível com tecnologias como RDF, SPARQL, OWL, SKOS. (W3C BRASIL, 2017).

É válido ressaltar que os termos “*Web* de Dados” e “*Web* 3.0” também se referem a *Web* Semântica, e o conceito em si é uma extensão da *web* atual.

Para atingir a meta de atribuir significados ao documentos postados na *web* Tim Berners-Lee propôs que se adicionasse informações descritivas em uma camada invisível dos arquivos, assim atribuindo significado semântico a elas, dessa maneira possibilitando que máquinas trabalhassem com os documentos em questão. Com esses avanços seriam possíveis consultas mais rápidas e ricas, melhores interações entre sistemas e uma maior interoperabilidade de dados.

No entanto, para que essa idéia seja realmente efetiva, é necessário que uma parte considerável do acervo digital tenha marcações para ligação dos dados. Berners-Lee (2006) relatou que técnicas de web semântica continuavam um recurso marginalmente utilizado. Apesar desse fato, ao longo dos anos o número de páginas

da internet com *Semantic Web Markup*[2] cresceu substancialmente, dando novo fôlego ao conceito de uma *web* com seus dados conectados.

Uma série de ferramentas e tecnologias livres são empregadas no desenvolvimento da *web* semântica, sendo algumas das mais proeminentes o modelo de dados de Ontologia, a metodologia de *Linked Data* (que emprega o formato RDF), dentre outros; Quando combinados estes recursos oferecem o arcabouço tecnológico que permite que visão de Berners-Lee seja possível, trazendo consigo todos os benefícios inerentes a ela.

2.3.1. Linked Data

Introduzido por Tim Berners-Lee (2006) a disciplina de *Linked Data*, ou Dados Conectados, trata de ilustrar uma nova forma de se publicar conhecimento e dados na internet. O termo *Linked Data* refere-se ao conjunto de melhores práticas para publicação e conexão de dados estruturados na *internet* (HEATH, 2009), sendo importante notarmos que este conceito originou-se de outro publicado por Berners-Lee: a *Web* Semântica. O autor ainda vai mais além e descreve *Linked Data* como "a *Web* Semântica feita de maneira correta" (BERNERS-LEE, 2008). Uma maneira de se interpretar a relação entre os dois termos, seria a de que a *Web* Semântica é composta por *Linked Data*, ou seja: a *Web* Semântica é o todo enquanto os dados conectados são as partes.

Dessa forma diz-se que a *Web* de Dados difere da *web* comum no fato de que a segunda navega através de *links* em documentos, enquanto a primeira se utiliza das relações para navegar em uma rede de conceitos (BERNERS-LEE et al., 2006). Algumas tecnologias sustentam essa ideia, para nomear uma parte, temos: URIs[3], HTTP e o formato de publicação RDF. Assim, enquanto a *web* convencional se utiliza de *hyperlinks*, *Linked Data* utiliza o padrão RDF para criar declarações sobre dados arbitrários, que sejam relacionados entre si.

Em um de seus documentos técnicos Berners-Lee (2006) estabeleceu quatro princípios para a publicação e conexão de dados sob sua arquitetura. Dessa forma, o autor esperava que um dia se atingisse um espaço virtual completamente interligado. Os princípios para publicação de *Linked Data* seriam:

1. Utilize URIs como nomes para seus recursos;
2. Utilize URIs HTTP para que seja possível que pessoas visualizem os recursos;
3. Quando um indivíduo visualizar o recurso, provenha informações úteis, utilizando os formatos padrões (SPARQL e RDF);
4. Inclua *links* para outras URIs, para que seja possível descobrir mais coisas.

[2] Elementos HTML semânticos (w3schools.com/html/html5_semantic_elements.asp).
[3] tools.ietf.org/html/rfc3986

Os princípios acima, muitas vezes apelidados de “princípios de *linked data*”, fornecem uma forma padrão para se publicar dados conectados na *web* clássica, pois aderem aos esquemas que vigoram no ambiente virtual. Através das tecnologias, modelos e princípios, discutidos aqui, torna-se possível publicar informações na *internet* de uma maneira mais inteligente, e que contribua para a iniciativa de uma rede de mundial computadores mais eficiente.

2.3.2. Linked Open Data

O conceito de *linked open data* (também conhecido como dados abertos conectados ou LOD) é muito similar ao de *linked data* convencional, contudo este último nem sempre precisa ser de livre acesso, enquanto o último exige tal qualidade.

Tim Berners Lee (2006) descreve LOD como sendo dados conectados publicados sob uma licença aberta[4] que não impeça sua reutilização livre de custo. Dessa maneira pode-se dizer que o campo de dados abertos conectados é onde *linked data* e dados abertos se encontram.

2.3.3. Ontologias

Durante muito tempo a palavra ontologia ficou confinada à esfera da filosofia, mas com o passar dos anos a computação passou a se utilizar do termo e conceito para uma miríade de segmentos, como o da *Web* Semântica. Gruber (1993) definiu uma ontologia como sendo a especificação de uma conceitualização, ou seja, uma descrição de conceitos e relacionamentos entre esses conceitos. Já Guarino (1998) elabora e expande a definição para: "Uma ontologia é uma lógica que explica o significado pretendido de um vocabulário formal, ou seja, o compromisso ontológico com uma conceituação particular do mundo.”. Baseado nessa definição, uma ontologia consistiria em conceitos, relações, e suas definições, propriedades e restrições representadas por axiomas. Uma definição mais atual, e relevante, foi dada pelo próprio Gruber:

> [...] uma ontologia define um conjunto de representações primitivas com as quais se modela um domínio de conhecimento ou discurso. As representações primitivas são geralmente classes (ou conjuntos), atributos (ou propriedades) e relacionamentos (ou relações entre os alunos). As definições das primitivas representacionais incluem informações sobre seus significados e restrições em sua aplicação logicamente consistente. (GRUBER, 2009).

[4] A licença MIT é um exemplo desse tipo de licença

Partindo de todas essas diferentes interpretações, podemos definir uma ontologia como um modelo de dados que representa um conjunto de conceitos (dentro de um determinado domínio) e seus relacionamentos, de uma maneira formalizada.

Os componentes mais comuns que formam uma ontologia, e o que representam, são:

- Indivíduos: instâncias ou objetos;
- Classes: coleções, conjuntos ou tipos de objetos;
- Atributos: características, propriedades ou parâmetros que objetos, e classes, podem possuir;
- Relações: as maneiras com que objetos podem interagir entre si;
- Restrições: descrições formais do que precisa ser verdadeiro para que uma asserção seja aceita;
- Regras: declarações em formato de sentenças se-então (antecedente-consequente).

O modelo de ontologias pode então ser utilizado para que humanos e máquinas comuniquem entre si conceitos de uma maneira mais concisa, levando ao estabelecimento um entendimento comum sobre o domínio do conhecimento que se está modelando, e seus relacionamentos em questão. Sendo assim, a *Web* Semântica acaba por depender de ontologias formais, para estruturar os dados subjacentes, com o objetivo de compreensão e transporte pelas máquinas (MAEDCHE; STAAB, 2001).

As vantagens trazidas pela modelagem de conhecimento através de ontologias são diversas - compartilhamento, reuso, confiabilidade, entre outras - para a área da computação, visto que dessa maneira nos é permitido trabalhar e gerenciar dados de áreas complexas, como: processamento de linguagens naturais, biologia molecular e bioinformática, para citar alguns campos. Um exemplo prático da aplicação de ontologias, que trouxe resultados surpreendentes, seria o Watson[5] da IBM. No artigo delineando o processo de sua criação, é citado que o Watson faz uso de ontologias como a DBPedia[6], em conjunto com diversas outras tecnologias, no processo de formação de suas respostas (FERRUCCI et al., 2010). Demonstrando assim a importância e potencial latente, presentes na estruturação, e conexão, das informações disponíveis no acervo do mundo digital.

2.3.4. RDF

De maneira a ser possível a criação da *Web* de Dados, era necessário que se criasse um *framework* para a modelagem de conhecimento de maneira apropriada, e que fosse compreendida por computadores. O *Resource Description*

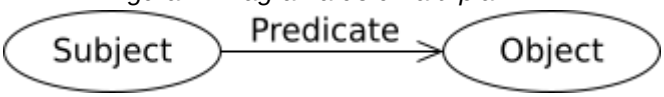
[5] ibm.com/watson

[6] wiki.dbpedia.org/about

Framework (comumente chamado de RDF) é um formato criado pela W3C. Originalmente desenvolvido como um modelo de metadados, o RDF foi adaptado para a modelagem de informações para uma *Web* Semântica, passando a integrar as recomendações da W3C em 1999. É importante ressaltar que o RDF apenas trata de modelar os dados e não de como apresentá-los, diferindo de outras linguagens da *web* clássica. Também devemos destacar a existência das diversas formas de serialização do formato, como, para exemplificar alguns, XML/RDF, N-Triplas, N-Quads, JSON-LD e RDFa, as quais tem suas vantagens e usos distintos.

O modelo de dados RDF é similar a modelagens mais conceituais, como entidade-relacionamento ou diagramas de classe, e é baseado na idéia de se fazer declarações sobre recursos utilizando expressões no formato sujeito, predicado e objeto, que são conhecidos como triplas. Cada componente da tripla é representado por um nodo RDF que é identificado por uma URI, sendo o objeto o único que pode assumir um valor literal ao invés de uma URI. Os nodos do predicado e objeto, dão significados ao nodo sujeito, sendo dever do nodo predicado estabelecer a relação entre o sujeito e objeto contidos na tripla (W3C, 2014). A dinâmica descrita pode ser melhor observada no diagrama da figura 2, que representa o grafo de uma tripla RDF.

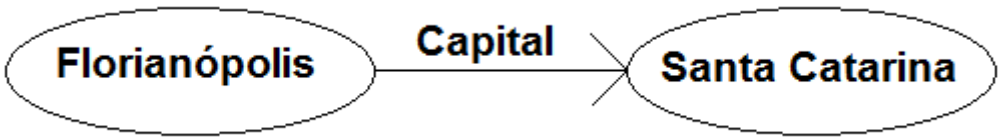
Figura 2: Diagrama de uma tripla RDF



Fonte: W3C (2014).

Desta maneira, se quiséssemos representar a frase “Florianópolis é a capital de Santa Catarina”, terminaríamos com uma estrutura com Florianópolis como sujeito, capital como predicado e Santa Catarina seria o objeto. A tripla resultante poderia ser representada no grafo da figura 3.

Figura 3: Grafo representando relação RDF.



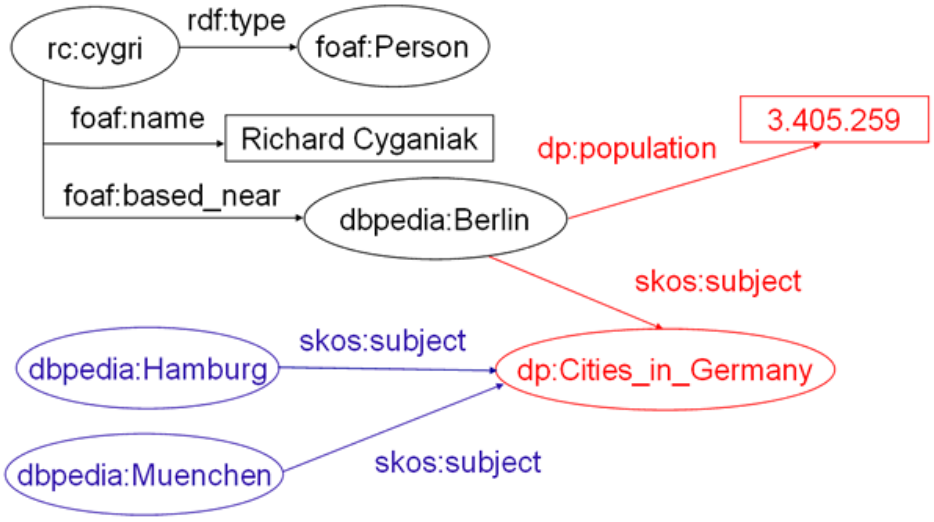
Fonte: Autor.

Em seu livro Heath e Bizer (2011) defendem que existem 2 tipos diferentes de triplas: literais e triplas de ligação RDF. O primeiro tipo teria um tipo literal (um número, string, data, entre outros) no nodo objeto, assim identificando uma propriedade do recurso - como, por exemplo, o nome, ou data de nascimento de uma pessoa. Enquanto isso, triplas de ligação RDF representam uma ligação entre dois recursos, criando uma maneira de se viajar entre estruturas RDF distintas através de sua relações no mundo real. Por exemplo: podemos utilizar essa arquitetura para ir de um recurso detalhando uma pessoa qualquer, para o de seu

cônjuge, pois existiria uma tripla de ligação explicitando a relação entre ambas as estruturas RDF.

Um entendimento mais concreto pode ser extraído ao observarmos a figura 4 (BIZER; CYGANIAK; HEATH, 2007), a qual representa o grafo criado para se representar as ligações RDF entre Berlim e outras cidades alemãs, bem como a conexão com um dos autores do artigo sendo discutido

Figura 4: Grafo representando os links RDF da cidade de Berlim.



Fonte: Bizer, Cyganiak e Heath (2007)

É possível representar recursos RDF através de URIs HTTP, e podemos perceber esse conceito na imagem acima, em razão de alguns sujeitos estarem representados por menções ao recurso *online* DBPedia, ao exemplo do nodo Berlim, enquanto alguns objetos são ilustrados por literais (população da cidade). De posse do conhecimento de que recursos que compõem triplas podem ser representados por URIs HTTP, podemos retornar ao exemplo simplificado da figura 3 e alterar sua estrutura para que passe a ser a seguinte:

- Sujeito: pt.dbpedia.org/resource/Florianópolis;
- Predicado: dbpedia.org/ontology/capital;
- Objeto: pt.dbpedia.org/resource/Santa_Catarina_(state).

No exemplo observamos o uso de uma ontologia para modelar o relacionamento entre a cidade e o estado, estabelecendo sua interrelação de maneira formal. Dessa forma podemos notar claramente as conexões sendo realizadas, e sendo possível representar associações utilizando URIs - um formato que computadores conseguem entender - notamos o quão viável é utilizar o formato RDF para uma *web* mais conectada, rica e com pesquisas mais rápidas e relevantes, do que as que são realizadas atualmente.

2.4. Dados Abertos

A *Open Definition*[7], organização que busca definir os princípios de "abertura" relativos à conhecimento (o que acaba por contemplar dados abertos), define como "abertos" todo tipo de conteúdo, sejam arquivos de mídia, documentos ou qualquer outro tipo de dado, que satisfaça alguns critérios em sua forma de distribuição, sendo eles os seguintes:

- Os dados devem estar disponíveis sob uma licença aberta;
- A obra deve estar disponível na íntegra, preferencialmente por *download*, a um custo razoável e acessível ao público;
- O conteúdo deve ser apresentado em um formato digital acessível (e que seja livre, ou legível por *software* aberto), que não dificulte o acesso com obstáculos tecnológicos desnecessários e que permita modificações;
- Qualquer pessoa deve poder acessar, utilizar, modificar e redistribuir os arquivos de forma livre (sujeito, no máximo, a requisitos que preservem a proveniência e a abertura).

Através desses parâmetros podemos notar o valor que a adoção do formato aberto por certas instituições, mais notavelmente as científicas e governamentais, proporcionaria à sociedade. Contudo algumas dessas áreas são relutantes em adotar o modelo. Segundo Janssen, Charalabidis e Zuiderwijk (2012) a resistência se dá pelo fato de que ao passar a publicar seus dados de maneira aberta, existe uma mudança na forma de governança, de uma fechada para aberta, que acarreta em um impacto substancial nos relacionamentos das agências emissoras, com o público que consome os dados abertos gerados por elas. Ademais os autores advertem contra os mitos de que a abertura dos dados criaria valor por si só - pois só são gerados benefícios se as informações forem utilizadas pelo público geral - e de que toda e qualquer informação deve ser compartilhada.

Mesmo assim, pesquisadores de outras áreas, como a científica, acreditam que caso seus colegas passassem a distribuir livremente seus dados, o campo como um todo só teria a ganhar, pois a prática leva a um aumento da cooperação, transparência e tende a desencorajar fraudes (MOLLOY, 2011).

2.4.1. Lei de Acesso à Informação e Transparência Ativa

Visando as vantagens trazidas pela publicação de seus dados, o governo brasileiro aprovou a Lei nº 12.527/2011[8], conhecida informalmente como Lei de Acesso à Informação, a qual entrou em vigor no dia 16 de Maio de 2012. A partir

[7] opendefinition.org

[8] Informações retiradas do portal de Acesso a Informação (acessoainformacao.gov.br/assuntos/conheca-seu-direito/a-lei-de-acesso-a-informacao).

desta data, todo e qualquer cidadão passou a poder requerer, sem necessidade de justificativa, os dados e informações de qualquer órgão ou entidade dos poderes públicos, nas esferas Federal, Estadual e Municipal.

Além disso a lei prevê a necessidade de antecipação dos pedidos dos cidadãos, por parte dos diversos órgãos governamentais, forçando-os a publicar seus dados na internet. Apesar de não usar o termo “dados abertos”, é exigido que as informações sejam publicadas em formatos não-proprietários e abertos, o que faz com que os dados sejam publicados livremente, de forma similar ao tema discutido aqui.

Um dos frutos dessa lei é o Portal Brasileiro de Dados Abertos[9], que foi construído por cidadãos e empresas da Sociedade Civil, em parceria com servidores públicos, que tinham interesse no conceito de Governo Aberto[10].

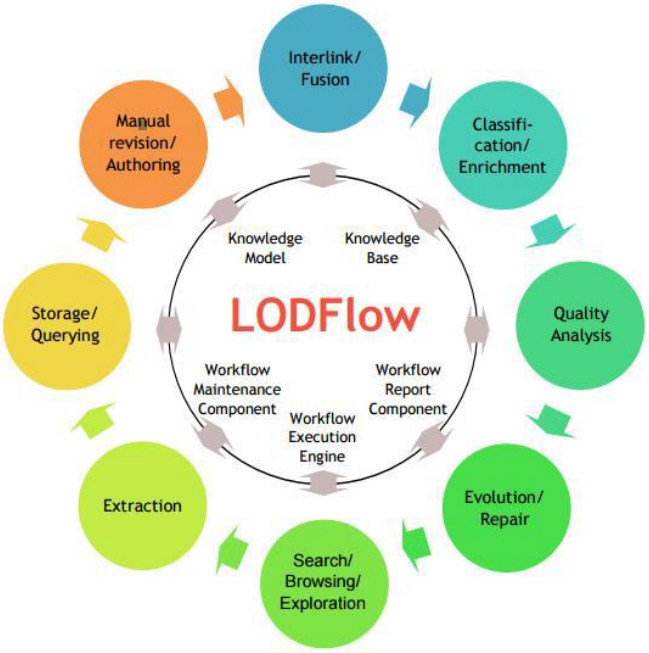
2.5. LODFlow

Grande parte do trabalho associado ao processo de *Linked Data* vem das tarefas relacionadas a preparação das informações para transformação para o formato RDF. Normalmente o processamento envolvido nesta etapa se dá através de *scripts* de aplicações proprietárias ou por fluxos de execução, e trabalho, manuais. Assim a extração e preparo das informações se torna extremamente custosa, em razão dos diversos passos exigidos - os quais devem ser feitos de maneira manual pelo operador dos instrumentos de software. Algumas dessas ações, não levando em conta a parte de disponibilização das informações, são: carga de dados brutos, extração e análise.

Como já discutido previamente, o *Linked Data Workflow Management System* (LDWMS), conhecido de maneira abreviada como *LODFlow*, é um programa que visa auxiliar no processo de preparação e publicação desse tipo de dado, conseguindo esse feito ao prover um ambiente propício para o planejando, execução, reutilização e documentação de *workflows* de dados conectados. Na figura 5 pode-se observar o ciclo de vida que é apoiado na implementação deste programa.

[9] dados.gov.br/
[10] opengovpartnership.org

Figura 5: Ciclo de Vida suportado pelo LODFlow.



Fonte: RAUTENBERG et al. (2015).

Para conseguir sustentar completamente o ciclo ilustrado, soluções para modelagem, documentação, registro, manutenção e criação de relatórios, são disponibilizadas. Dessa maneira é possível reunir a maior parte das necessidades básicas, dos usuários que buscam publicar *Linked Data*, em uma única aplicação.

2.6. SysLODFlow

Como já discutido anteriormente, todo o processo de publicação e manutenção de dados conectados exige uma miríade de conhecimentos relacionados ao processo de *Linked Data*, bem como acerca de ferramentas de *software*, para que ao final se tenha um *dataset* de qualidade e que seja utilizável para *upload* à *internet*. O advento do LODFlow trouxe a possibilidade de se suportar todo o ciclo de vida de conjuntos de *Linked Data*, em um único artefato de *software*. Para isso são disponibilizadas componentes específicos que dão suporte aos diferentes estágios do processo, assim permitindo que a ferramenta seja utilizada na criação de conjuntos ou manutenção dos mesmos.

Contudo, a carga ao usuário ainda é elevada, visto que o LODFlow demanda tempo considerável para que seja dominado, exigindo tanto conhecimentos específicos quanto trabalho para processamento das informações. Motivados pela alto esforço e conhecimento, Morais e Morais (2016) decidiram criar uma aplicação *web*, batizada de SysLODFlow, que facilitasse o processo como um todo. Em suas

palavras eles desenvolveram uma “ferramenta gráfica customizada para registro e manipulação das instâncias das classes envolvidas no LODFlow em sua base de conhecimento, a ontologia LDWPO[11]. O objetivo do SysLODFlow acabou por ser: criar um ambiente customizado para registros de projeto, *workflows*, passos e execuções, para, deste modo, melhorar a experiência do usuário final ao facilitar o entendimento dos processos envolvidos com a manipulação de ontologias, através do encapsulamento de todos os conceitos necessários para operação da LDWPO na aplicação desenvolvida.

Segundo os criadores da ferramenta, a escolha por se criar um aplicativo *web* se deu por alguns fatores chave: facilidade de acesso pelos computadores atuais, compatibilidade garantida independente do SO do usuário, atualização de versões facilitada e uma maior propensão de integração com os componentes do LODFlow (MORAIS; MORAIS, 2016).

O funcionamento da aplicação se dá através da criação de projetos de publicação de *Linked Data*, os quais contêm *workflows* que podem ser executados quando necessário. Os *workflows* em si são compostos por operações chamadas *steps*, que formam o fluxo de execuções que compõem o processo. Ao final da execução de um *workflow* dois artefatos são gerados: um relatório contendo os resultados da execução e um conjunto de dados RDF.

Todos componentes descritos podem ser salvos em uma base de dados, para que sejam recuperados e executados (ou alterados) em um momento posterior, de maneira a oferecer dinamismo ao procedimento de publicação e favorecendo o reuso de *workflows* de execução.

Outros aspectos técnicos da aplicação envolvem o fato de que a mesma deve rodar em um servidor Linux e que a linguagem de programação escolhida para sua criação sendo o Java. O *software* também foi desenvolvido seguindo uma estrutura em 3 camadas, as quais podemos chamar de: cliente, aplicação e fonte de dados (LIU; HEO; SHA, 2005), além da nomenclatura adotada originalmente. Abaixo o *stack* de tecnologias empregadas pode ser observado na figura 6.

[11] github.com/AKSW/ldwpo

Figura 6: Esquema representando as camadas de aplicação e seus componentes.

Fonte: Morais e Morais (2016).

3. PROCEDIMENTOS METODOLÓGICOS

Este capítulo trata sobre a abordagem tomada para a execução das tarefas necessárias, para se atingir os objetivos descritos anteriormente neste documento.

O restante desta obra é dividida em dois capítulos, que discorrem, respectivamente, sobre: o estado inicial e trabalho realizado sobre o SysLODFlow e sobre as conclusões e resultados que se chegou.

Sendo assim, na divisão a seguir serão apresentados os processos de diagnóstico do aplicativo, desenvolvimento de correções e melhorias realizados com esse objetivo em mente. Da mesma forma, após detalhar-se as atividades realizadas, estas serão evidenciadas e abordadas em um capítulo novo com os resultados alcançados, idéias para trabalhos futuros e conclusões obtidas.

Primeiramente o processo de diagnóstico será trabalhado com o intuito de providenciar uma visão sobre o estado geral em que se encontrou o SysLODFlow, um entendimento da implementação dos casos de uso e requisitos, bem como as conclusões que se chegou após o estudo do código fonte, testes manuais e análise de interface.

Após a descrição geral da aplicação, serão expostos, de maneira mais pontual e aprofundada, os pontos em que existem falhas no design e/ou funcionamento do *software*. Soluções que podem ser adotadas para se corrigir a deficiência encontrada serão apresentadas e discutidas. Também poderão ser apresentadas propostas melhorias possíveis das funcionalidades, que devem ser implementadas da estaca zero pois não estavam inclusas no escopo do projeto original, ou que melhoram algum aspecto já implementado do sistema.

Em seguida serão selecionados os pontos que se julgou serem prioritários para o bom funcionamento da aplicação, e as melhores soluções encontradas serão desenvolvidas - seguindo os conceitos de engenharia de *software* e boas práticas

de desenvolvimento. Os tópicos levantados que não forem selecionados para desenvolvimentos servirão como recomendações trabalhos futuros.

Finalizando-se o processo de desenvolvimento um pedido de *merge*, no projeto original, deve ser realizado ao repositório GitHub[12] que hospeda o SysLODFlow, para que a aplicação possa ser acessada pelos interessados em aprender sobre ela, ou até mesmo contribuir para o desenvolvimento.

Por último o capítulo seis apresentará todos os resultados provenientes da parte de desenvolvimento, as conclusões que se chegou após a análise do antes e depois do SysLODFlow e as idéias de trabalhos futuros que podem ser desenvolvidas, para que a ferramenta se torne mais completa e útil, culminando assim no encerramento deste projeto.

4. DIAGNÓSTICO E APERFEIÇOAMENTO DO SYSLODFLOW

Como descrito anteriormente, esta parte da obra tratará dos processos de diagnose e evolução do código do SysLODFlow. Para isso o capítulo está dividido nos quatro segmentos abaixo:

1. Visão Geral do Estado do Sistema;
2. Problemas e Erros Encontrados;
3. Possíveis Melhorias;
4. Desenvolvimento das Melhorias e Correções do Projeto.

Seguindo essa organização de texto e leitura, espera-se que seja possível entender a condição em que o projeto foi encontrado, quais são seus problemas, como é possível saná-los e o que foi desenvolvido e aplicado.

4.1. Visão Geral do Estado do Sistema

Antes de se iniciar o processo de desenvolvimento das expansões das funcionalidades do SysLODFlow, era primeiramente necessário entender os seguintes fatores:

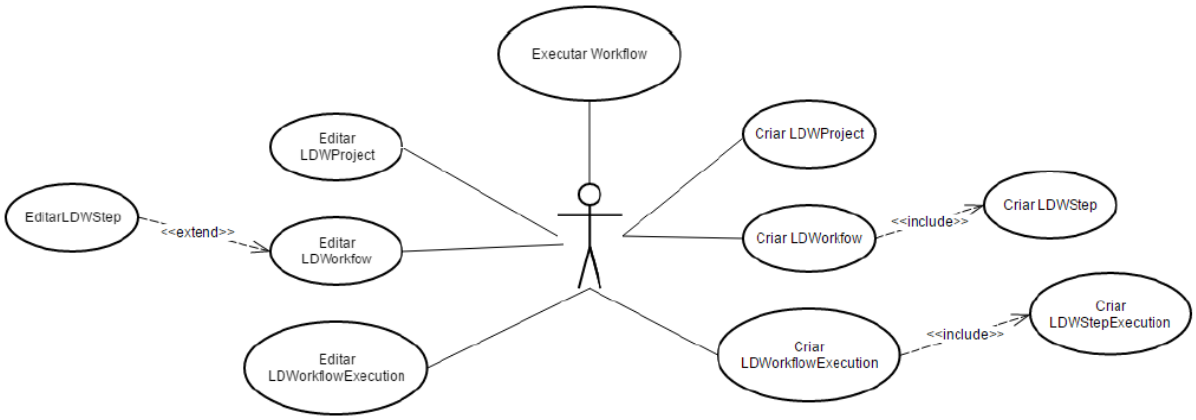
- O que a ferramenta almeja realizar;
- Como foram implementadas as funcionalidades;
- Como realizar as melhorias desejadas.

[12] github.com/bruno-edo/syslodflow_01

Para sanar a dúvida do primeiro ponto se recorreu ao documento de TCC que deu origem a ferramenta (MORAIS; MORAIS, 2016). No documento o processo de elaboração e amparo teórico é explicado pelos autores, o que acabou por dar uma boa base para entendimento do que se queria alcançar. Durante esta etapa, compreendeu-se o desejo dos autores de se utilizar da ontologia LDWPO, para automatizar o processo de publicação de dados abertos conectados, como demonstrado por Rautenberg et al. (2016) em seu trabalho.

Além disso, o documento explica diversas decisões de design da ferramenta, bem como o funcionamento de certos aspectos da mesma. Na figura abaixo pode-se notar o diagrama de casos de uso do SysLODFlow, o qual ilustra as operações que o aplicativo deve poder realizar. Com esse diagrama podemos começar a visualizar o que deveria ser feito pelo aplicativo, e ter uma idéia básica de como cada funcionalidade interage com as outras.

Figura 7: Casos de uso do SysLODFlow.



Fonte: Morais e Morais (2016).

Entretanto, uma melhor documentação do projeto é inexistente, os documentos que elaboram sobre os aspectos funcionais e teóricos do projeto se limitam ao relatório de TCC e artigos citados. Apesar de ambos serem boa fonte de informação, o objetivo desses não é dar um entendimento completo sobre a implementação dos recursos de *software* selecionados para o desenvolvimento, ou sobre a estruturação do código fonte do programa. Dessa maneira, fica aparente que uma documentação voltada para desenvolvedores não está presente nos arquivos do projeto, significando que uma das etapas do ciclo de desenvolvimento de *software* - a de documentação dos trabalhos realizados - foi ignorada. Além disso, ficou claro que seria necessária a realização da engenharia reversa das funcionalidades, a fim de se resgatar o design dos casos de uso já descritos na figura 7, compreender as operações de uso geral e desvendar o processo de instalação da ferramenta.

Como já mencionado em uma seção anterior, o cenário encontrado é clássico na área de engenharia reversa de *software*: deseja-se atualizar um programa/módulo, mas a documentação do mesmo é defasada e/ou inexistente. No caso ilustrado aqui pode-se dizer que a documentação é nula, já que não se pôde

compreender muito sobre o código desenvolvido com base apenas nos documentos acadêmicos que se tinha a disposição.

Seguindo a linha das três perguntas chaves que a disciplina de engenharia reversa tenta responder, podemos ver que a primeira é respondida pelo diagrama de casos de uso, assim nos deixando com mais duas questões pendentes.

A interrogação seguinte é, provavelmente, a mais crucial e complexa do processo: como representações de alto nível podem ser obtidas através de traços de execução e arquivos de código fonte. Para auxiliar nessa etapa de análise do código, foram utilizadas ferramentas de criação e *debug* de *software* Java[13], uma delas sendo a IDE Eclipse (ECLIPSE FOUNDATION, 2017). O ambiente proporcionado pelo Eclipse contava com todas as ferramentas necessárias para se iniciar o estudo, desde o suporte a linguagem Java, até *plugins* para criar um servidor de aplicação para se rodar a instância do programa.

No entanto, o primeiro desafio foi encontrado ao se tentar instalar corretamente o SysLODFlow. Erros provenientes do servidor de aplicação acusavam a falta de certas configurações, desconhecidas até o momento. Para resolver o impasse encontrado foi necessário entrar em contato com os autores do programa, somente dessa maneira foi possível compreender o que faltava ser configurado e colocar o SysLODFlow no ar.

De posse dos recursos citados, com a ferramenta instalada e funcionando, finalmente foi possível começar a compreender e traçar o funcionamento interno do SysLODFlow, através da exploração da interface gráfica, observação da execução do código fonte e das informações obtidas através de seu *debug*.

Durante os primeiros contatos com a ferramenta, tentou-se realizar os casos de uso previstos através da interface *web*. Percebeu-se durante esses procedimentos que muitas funcionalidades esperadas aparentavam não estar funcionando, não terem sido implementadas ou, apesar de implementadas, não estarem finalizadas. Muitas dessas suposições foram confirmadas ao se verificar as mensagens apresentadas pelo console de execução da IDE, bem como diretamente pelo código fonte, como pode ser observado na figura abaixo, a qual representa um botão sem funcionalidade atrelada.

Figura 8: Código do botão de download com propriedade de ação não implementada.

```
<h:form id="form-download-owl">
    <h:commandButton id="downloadOwl"
    action="#" styleClass="btn btn-success btn-xs"
    value="#{msg['crud.file.download']}" />
</h:form>
```

Fonte: Autor.

Além do mais, certas operações e configurações pareciam terem sido implementadas de maneira *hard coded*, o que vai de encontro com o propósito original da concepção do SysLODFlow.

[13] www.java.com/en/

A descoberta mais grave relacionada a esse tópico, se deu quando, uma chamada de execução a um programa Java externo foi descoberta, com seu endereço de localização apontando para uma pasta da máquina de um dos criadores do sistema (ilustrado na figura 9). Tal fato fazia com que, o fluxo de execução não fosse passível de reprodução fora do ambiente de desenvolvimento original, significando que parte essencial do aplicativo - a execução de *workflows* - não funcionava.

Figura 9: Função com endereço de arquivo hardcoded para máquina do autor original.

```
public void doExecute() {
    try {
        Runtime.getRuntime().exec("java -jar C:\\Users\\Jhonatan\\Downloads\\BancoDeClubes\\"
                                  + "BancoDeClubes\\BancoDeClubes\\dist\\BancoDeClubes.jar");
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Fonte: Autor.

Diversos erros mais foram encontrados durante a etapa de análise do código fonte, fazendo com que ficasse claro o estado de imaturidade do projeto como um todo e de como a premissa de uma ferramenta funcional estava incorreta. Assim sendo, percebeu-se a necessidade de implementar e corrigir, as funcionalidades básicas que se supunha estarem operantes.

A barreira imposta pela necessidade da engenharia reversa vai de encontro com os ideais de *software* aberto com os quais se construiu o SysLODFLow. Além do mais, após o terceiro questionamento da engenharia reversa ser considerado, notou-se a necessidade de se elaborar meios de passar adiante as descobertas realizadas, pois tais informações são essenciais a indivíduos que desejam contribuir para a evolução do aplicativo, mas só são claras após a análise minuciosa do código e de sua execução. Para que, dessa maneira, novos contribuidores do projeto não tenham que realizar novamente o processo de engenharia reversa apresentado aqui.

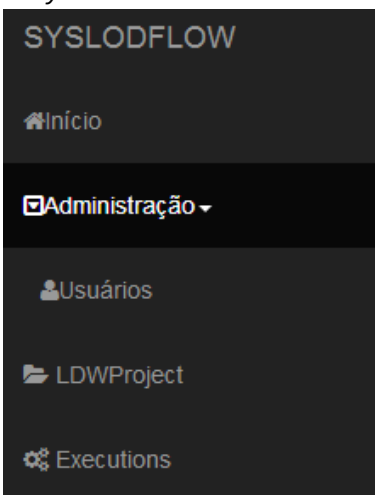
4.2. Problemas e Erros Encontrados

Esta seção trata dos erros encontrados durante a avaliação da ferramenta, e de seu código fonte. São considerados erros os problemas que impeçam a utilização pretendida da ferramenta, gerem exceções no código e/ou vão contra regras e padrões de desenvolvimento para o desenvolvimento de aplicações *web*.

4.2.1. Layout Geral das Páginas do Sistema

Diversos componentes visuais, como ícones e botões, estão mal alinhados, ou excedendo a largura de seus contêineres. Um exemplo disso pode ser observado na figura 7, onde se pode notar os ícones sobrepondo texto do menu principal e dificultando a leitura.

Figura 10: Menu SysLODFlow exibindo ícones desalinhados.



Fonte: Autor.

Tal fato vai contra as regras para composição de interfaces gráficas, bem como os conceitos de UX. Sendo assim, é preciso que se corrijam os erros identificados, fazendo com que a aplicação seja consistente nos três principais *browsers* da atualidade: Google Chrome, Firefox e Safari.

Uma das possíveis causas para o mal posicionamento desses artefatos gráficos, são erros (ou limitações) na aplicação de classes CSS. Como o aplicativo se utiliza do *framework* Bootstrap[14], é possível que atualizá-lo da versão 3, utilizada no projeto, para a iteração 4, seja a melhor opção para sanar alguns dos erros observados. Contudo, o *framework* sofreu alterações drásticas entre as versões 3 e 4, sendo assim atualizar o Bootstrap pode vir a quebrar a formatação das páginas - o que acarretaria em um processo oneroso e que talvez não seja possível devido às constrições de tempo desta obra. De qualquer maneira, ao se atualizar a biblioteca Bootstrap deve se fazer uso do guia de migração oficial (OTTO; THORNTON, 2017). Durante o processo de migração é possível que outros *frameworks* e/ou bibliotecas precisem ser atualizadas em conjunto com o *Bootstrap*.

Ademais, múltiplas páginas seguem padrões diferentes de design, acarretando na não existência de regras para a composição das interfaces. Tal ocorrência faz com que não vigore uma lógica de navegação padrão, fazendo com que a carga cognitiva para operação do sistema seja maior, assim prejudicando a experiência de operação do usuário.

[14] getbootstrap.com/

É necessário que as configurações das páginas do sistema sejam simples e consistentes entre si, já que esse são princípios chave da disciplina de UX (KIM, 2013). Ao serem aplicados, esses conceitos acabarão por criar uma identidade visual para o SysLODFlow - a qual outros desenvolvedores devem adotar, ao elaborar funcionalidades que exijam telas novas.

4.2.2. Botões Apresentando Diversos Comportamentos Errôneos ou Não Funcionando

Durante o processo de avaliação da interface gráfica, foi possível observar diversos aspectos do funcionamento de componentes gráficos que não são condizentes com práticas de desenvolvimento efetivo. Os mais preocupantes vieram da forma com que as ações desencadeadas, e verificações de segurança, de certos botões foram implementadas. Também foram notados casos onde certos botões não realizam as funções para as quais foram designados.

Abaixo pode-se ler a lista com descrições detalhadas dos problemas mais significativos relacionados a botões:

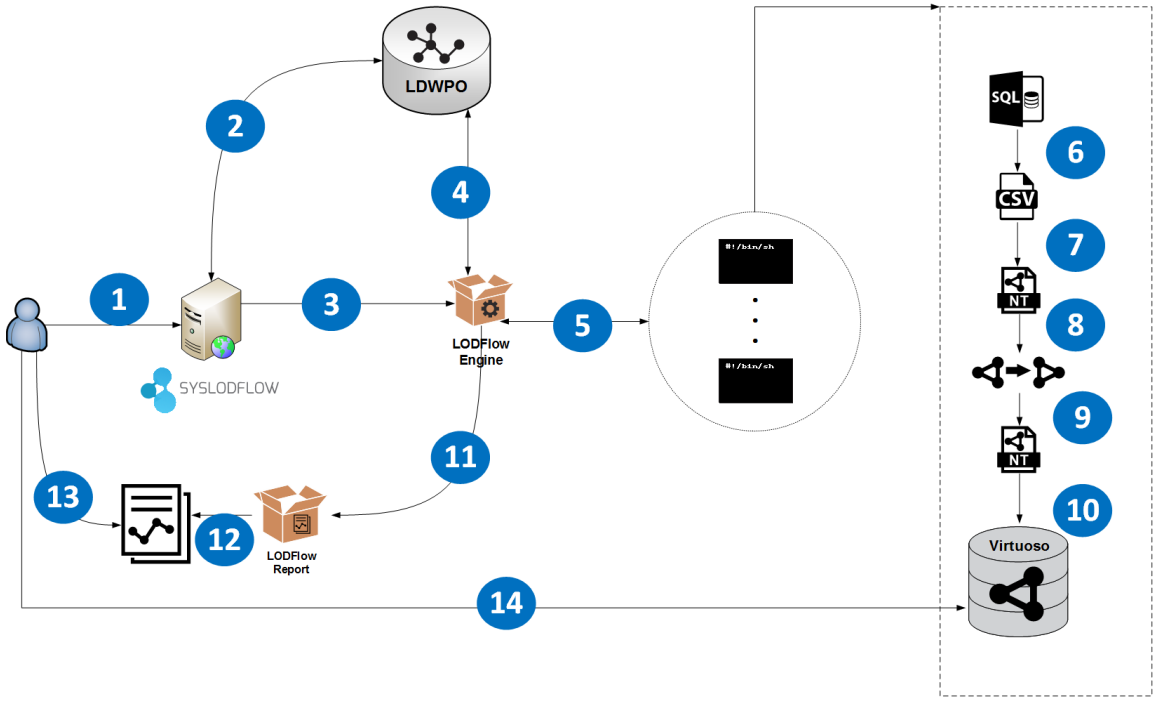
- Botões “Salvar” e “Voltar”, de certas páginas, estão realizando verificações em campos fora de seu escopo. Este fato faz com que só seja possível retornar/salvar quando todas as áreas de *input* de um formulário estejam preenchidas corretamente;
- Botões para exclusão de arquivos não foram implementados;
- Botões para *download* de arquivos não foram implementados.

Como se pode notar, existem diversos comportamentos anômalos associados a esses botões - alguns deles criando *bugs* de alta severidade, outros impedindo o uso de funcionalidades do sistema. Assim se faz necessária a adoção de diversas correções para sanar-se todos os erros apresentados.

4.2.3. Processo de Criação e Execução de LDWProjects e Arquivos de Ontologia

Quando o SysLODFlow estava sendo concebido era necessário criar um ciclo de operação para publicação dos dados conectados. O que se fez foi utilizar o exemplo da Qualis Brasil - desenvolvido por Rautenberg (2015) como prova de conceito para o projeto da ontologia LDWPO. Nesse primeiro caso protótipo os passos (e outras dependências, como configuração de ferramentas de apoio) para extração, transformação e publicação dos dados são guardados em um arquivo da LDWPO e interpretados pelo LODFlow para se criar um ciclo de publicação que seja reutilizável. O circuito criado para o SysLODFlow, baseado nessas idéias, pode então ser observado em sua totalidade abaixo, na figura 11.

Figura 11: Ciclo de operação desenvolvido para o SysLODFlow.



1. Usuário acessa SYSLODFLOW através de seu navegador, realiza as operações devidas e mandar executar o fluxo.
2. SYSLODFLOW armazena dados na ontologia
3. SYSLODFLOW chama a execução do LODFlow Engine para execução do fluxo informando o fluxo a ser executado
4. LODFlow Engine recupera da ontologia os dados relativos ao workflow a ser executado
5. LODFlow Engine realiza chamadas via script para execução de ferramentas de apoio, configuradas nos passos
6. Dados são extraídos de um banco de dados relacional (MySQL) e convertidos para CSV
7. Sparqlify realiza o mapeamento dos dados contidos no CSV e converte para triplas em formato NT
8. Arquivos de triplas é armazenado no banco de dados Virtuoso
9. Triplas são recuperadas do virtuoso e através do LIMES ligadas ao DBPedia
10. Triplas já ligadas ao DBPedia são armazenadas novamente no triple store Virtuoso
11. LODFlow engine aciona LODFlow Report para geração de relatório de execução do workflow
12.LODFlow Report gera relatório em HTML contendo informações acerca da execução
13. Usuário pode realizar o acesso ao relatório em HTML para verificar status da execução
14. Usuário pode realizar consultas sobre o banco de dados de triplas Virtuoso através de seu endpoint SPARQL

Fonte: Morais e Morais (2016).

O que se pretendia então era usar o projeto da Qualis, que já se sabia que funcionava e era passível de reprodução, para validar o ciclo do SysLODFlow. Contudo diversos erros e equívocos foram realizados durante a implementação dessa parte do sistema, o que acabou por fazer com que a execução de *workflows* não funcione.

Quando o aplicativo foi desenvolvido seus criadores (MORAIS; MORAIS, 2016) decidiram usar uma ontologia já populada como modelo para a criação de novos arquivos desse tipo, no caso foi usada a ontologia do caso de exemplo da Qualis Brasil, dessa maneira apenas pontos específicos do documento são alterados quando se cria novos projetos. Além disso a sintaxe LDWPO foi modificada por eles, com intuito de se localizar mais facilmente as informações de projeto e *workflow*. Essas remodelações podem ser observadas nas figuras a seguir, as quais mostram a descrição de um projeto na LDWPO original e nas ontologias geradas pelo SysLODFlow.

Figura 12: Projeto instanciado na LDWPO.

```
<owl:NamedIndividual rdf:about="http://ldwpo.aksw.org/terms/1.0/project_QualisBrasil">
        <rdf:type rdf:resource="http://ldwpo.aksw.org/terms/1.0/LDWProject"/>
        <goal rdf:datatype="&xsd;string">Offering Qualis Periodical Index according to
        the principles Linked Open Data</goal>
        <name rdf:datatype="&xsd;string">QualisBrasil</name>
        <description rdf:datatype="&xsd;string">QualisBrasil is part of http://lod.unicentro.br
        endpoint. It aims to aid researchers to collect Linked Open Data about Qualis Index in
        bibliometrics or scientometrics studies. Available data:
        Periodical ISSN, Periodical Name, Knowledge Field, Qualis Index, and Year.</description>
        <homepage rdf:resource="http://ldwpo.aksw.org/terms/1.0/homepage_QualisBrasil"/>
        <ldWorkflow rdf:resource="http://ldwpo.aksw.org/terms/1.0/ldWorkflow_maintaining_qualisBrasil"/>
        <creator rdf:resource="http://ldwpo.aksw.org/terms/1.0/person_sandro_rautenberg"/>
        <report rdf:resource="http://ldwpo.aksw.org/terms/1.0/report_QualisBrasilProject"/>
</owl:NamedIndividual>
```

Fonte: (RAUTENBERG, 2015).

Figura 13: Projeto instanciado na ontologia gerada pelo SysLODFlow.

```
<LDWProject rdf:about="http://ldwpo.aksw.org/terms/1.0/ldwProject_testedois">
  <report>
    <Report rdf:about="http://ldwpo.aksw.org/terms/1.0/report_testedois">
      <location>
        <Location rdf:about="http://ldwpo.aksw.org/terms/1.0/location_report_testedois">
          <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >C:\ldwProjects\\testedois</value>
        </Location>
      </location>
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >report_testedois.html</name>
    </Report>
  </report>
  <homepage>
    <Homepage rdf:about="http://ldwpo.aksw.org/terms/1.0/homepage_testedois">
      <location>
        <Location rdf:about="http://ldwpo.aksw.org/terms/1.0/location_homepage_testedois">
          <value rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
          >a</value>
        </Location>
      </location>
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >homepage_testedois</name>
    </Homepage>
  </homepage>
  <creator>
    <owl:NamedIndividual rdf:about="http://ldwpo.aksw.org/terms/1.0/person_dbpedia_creator">
      <name rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
      >University of Leipzig</name>
```

Fonte: Autor.

As diferenças na estrutura dos arquivos fazem com que as ontologias geradas pelo SysLODFlow não sejam compatíveis com o LODFlowEngine[15], fazendo com que não seja possível executar os passos descritos na figura 11. Para corrigir esse equívoco de implementação é necessária fazer com que uma ontologia LDWPO “virgem” (só contendo termos gerais como o que é o fermato CSV, por exemplo) seja usada como modelo. Também é essencial que o formato correto da LDWPO não seja alterado, para preservar sua compatibilidade com o LODFlowEngine e LODFlowReport.

[15] O LODFlowEngine é a parte do LODFlow responsável por executar *workflows* de projetos instanciados na ontologia LDWPO.

Com a modificação das ontologias geradas pelo sistema, se faz imprescindível reprogramar toda a parte de leitura e escrita das informações em arquivos de ontologia. Isso acarreta na reestruturação de boa parte do código da aplicação, visto que todas as informações acessadas pelo SysLODFlow ficam guardadas em disco nos arquivos de ontologia, fazendo dessa uma empreitada significativa e que demandará um bom tempo de execução para ser completada.

Outro equívoco cometido no desenvolvimento inicial foi o entendimento de que objetos do tipo LDWorkflowExecution representavam instância de execução de projetos, quando em realidade estes contém os status e mensagens oriundos das execuções de *workflows* pertencentes a LDWProjects. Ademais é preciso ressaltar o fato de que o método responsável por essa parte vital do funcionamento aparenta não ter sido codificado corretamente, pois esse apresenta uma chamada a um programa externo desconhecido, como evidenciado na figura 9. Assim se faz necessário corrigir a seção de execução de *workflows* para que os objetos corretos passem a ser enviados ao LODFlowEngine e as tarefas cruciais do ciclo de publicação sejam efetuadas.

Além de tudo que já foi discutido até este ponto existem outros erros menores que precisam ser endereçados:

- *Upload* de arquivos de configuração para as ferramentas de suporte não foi implementado;
- Métodos e telas para *upload* de arquivos de *dataset* CSV não foram implementados;
- Métodos para execução de SQLs para extração de *datasets* de BDs remotos não foram implementados;
- A criação de *scripts Shell* apresenta erros que fazem com que os comandos gerados não sejam executados corretamente;
- Ferramentas necessárias, como o LIMES e LODFlowEngine, não foram incluídas nos arquivos do projeto.

Os problemas gerados por esses pontos devem ser corrigidos de maneira adequada, a fim de se criar as condições para o funcionamento preciso do SysLODFlow.

4.2.4. Arquivos XHTML Com Erros de Sintaxe

Certos arquivos XHTML apresentam erros de sintaxe, o que faz com que as páginas geradas por eles seja completamente inacessível, ou tenha erros visuais, ou de interação.

Torna-se necessária a correção do código, para que as páginas possam funcionar da maneira que foram arquitetadas.

4.2.5. *Strings* Com Erros

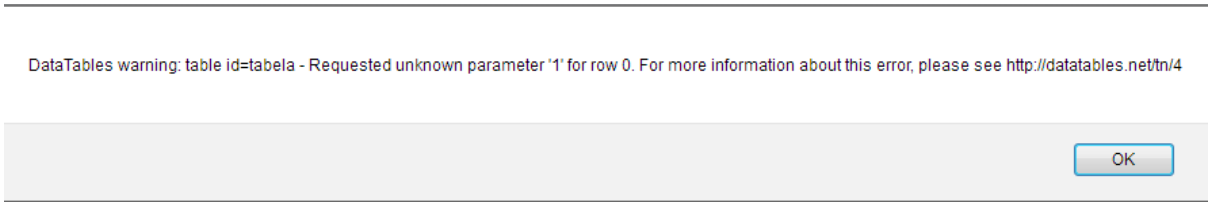
Algumas *strings* apresentam erros de escrita, ou são construídas de maneira errada.

As instâncias de texto que não são formatadas, ou escritas corretamente devem ser corrigidas, para que o texto apresentado ao utilizador seja claro e livre de ambiguidades.

4.2.6. Exceções Não Tratadas

Atualmente a aplicação apresenta diversos erros não tratados corretamente, ou seja: não existem indícios, na forma de mensagens contextualizadas e informativas, de que um erro foi detectado ou o tratamento é simplesmente inexistente, como pode ser visto na figura 14.

Figura 14: Exceção não tratada pelo SysLODFlow.



Fonte: Autor.

O erro específico exibido na figura acima é observado ao se iniciar a aplicação pela primeira vez, ao se tentar acessar a página de projetos. A falha é ocasionada pois o aplicativo tenta carregar e exibir os projetos cadastrados, dos quais ainda não existem instâncias no disco rígido.

Outras instâncias de exceções podem ser observadas ao e utilizar o sistema por algum tempo. As mesmas devem ser propriamente tratadas, e mensagens condizentes devem ser apresentadas ao usuário, para que o mesmo possa reportar ou, dependendo do caso, corrigir o erro no seu fluxo de utilização do SysLODFlow.

4.2.7. Classe Usuário Com Propriedades Conflitantes

A classe “Usuario” apresenta como uma de suas propriedades mapeadas um endereço de e-mail. Contudo, durante o procedimento de cadastro, de um novo usuário no sistema, não é apresentado campo ou opção para se informar tal endereço. Ademais, na tabela do BD relacional, um campo para armazenar essa informação não é criado. Sendo assim, um erro é encontrado quando se tenta acessar as informações do operador logado, o que impede de que essas sejam visualizadas e alteradas, caso se deseje.

Para corrigir este erro, é necessário adicionar um campo (com os devidos tratamentos) à página de cadastro, e edição, de usuário, bem como modificar o

script de criação da tabela no banco de dados, para que essa passe a ter um campo destinado a esse dado.

De outra maneira, deve-se remover a propriedade e-mail, e qualquer referência a ela nos arquivos, da classe “Usuario”.

4.2.8. Projetos Novos Não São Separados Em pastas Distintas

Todos os arquivos de ontologias do LDWProject são salvos na mesma pasta em disco: C:/syslodflow (em sistemas *Windows*), enquanto arquivos relacionados ao projeto ficam em uma pasta, de mesmo nome do projeto, no diretório C:/ldwProjects.

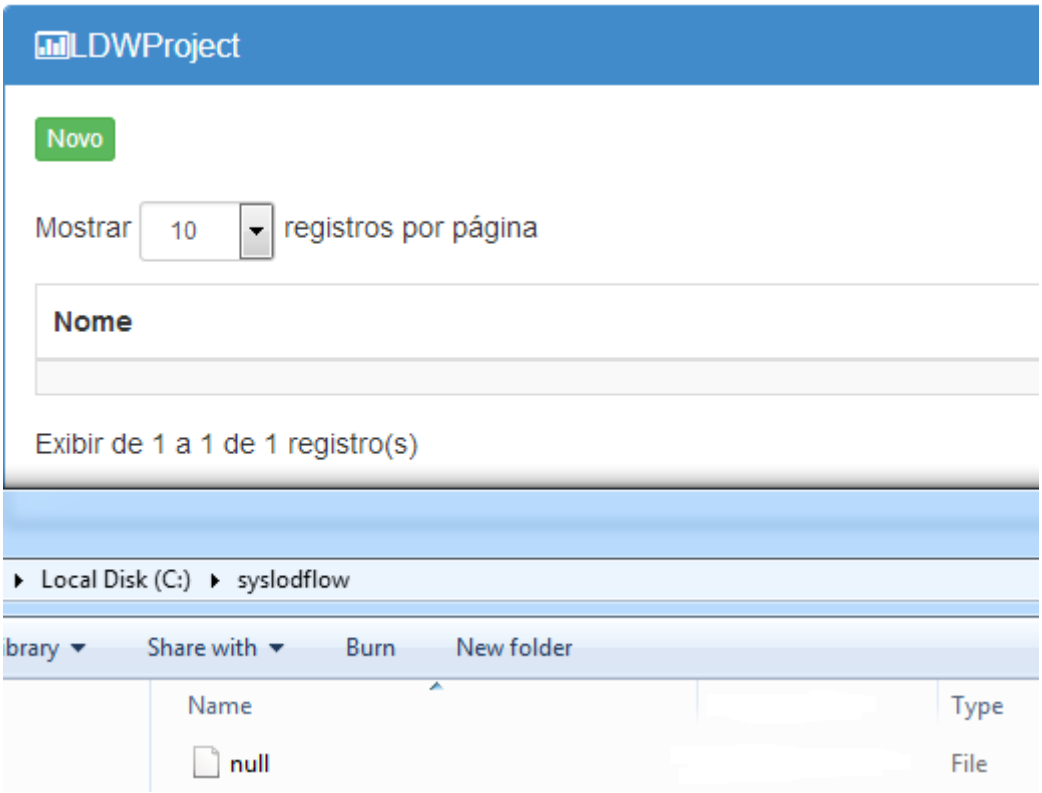
O comportamento correto seria separar os projetos em pastas diferentes, de maneira que também se segregue os arquivos de usuários diferentes.

Uma possível correção para este problema seria criar pastas para cada utilizador e, dentro dessas, criar pastas para cada projeto registrado pelo mesmo. A aplicação desse processo seria necessária tanto para os arquivos de ontologia, quanto para os relacionados ao projeto (*scripts*, documentos de entrada e configuração de ferramentas auxiliares).

4.2.9. Documentos de Projeto Sendo Criados Mesmo Que o Processo Não Tenha Sido Completado Corretamente

Durante o processo de registro de novo LDWProject existe a possibilidade, desde que certas condições sejam satisfeitas, de se criar arquivos nulos. Ou seja, projetos que não foram registrados passam a ser exibido pelo sistema, apresentando, também, conteúdos que não estão armazenados em disco.

Figura 15: Arquivo de projeto nulo e interface gráfica da seção de LDWProjects.



Fonte: Autor.

Caso outro projeto (dessa vez tendo suas informações devidamente preenchidas) seja salvo, o arquivo nulo passa a ser exibido sob o nome de QualisBrasil (como ilustrado na figura abaixo). É importante ressaltar que se por acaso a página de execuções for visualizada e o projeto QualisBrasil selecionado, serão exibidos registros de relatórios não existentes em disco. Isso se deve ao fato de um arquivo de ontologia *template*, o qual apresenta ligações para os relatórios citados, ser utilizado na geração de novos cadastros.

Figura 16: Projeto nulo sendo exibido sob nome de QualisBrasil.

Nome	Criador
Teste	University of Leipzig
QualisBrasil	Ivan Ermilov

Exibir de 1 a 2 de 2 registro(s)

Fonte: Autor.

O erro apresentado aqui se deve ao fato de que, se todos os campos de criação de novo projeto forem preenchidos corretamente, ao se inserir um novo nome de autor, o arquivo de projeto será salvo em disco com suas informações parcialmente preenchidas. Esse comportamento foi adicionado em virtude do fato de

todos os possíveis nomes de autores para projeto serem salvos nos arquivos de configuração.

A primeira vista podemos imaginar que o *bug* discutido aqui é proveniente do botão “Salvar”, da tela de autores, quando em realidade o verdadeiro erro está no fluxo de criação de arquivos e na escolha do local para armazenamento das informações de autores.

Uma correção efetiva para o problema descrito seria, fazer com que o botão “Salvar” do cadastro de autores pare de gerar arquivos, transferir os nomes registrados para uma tabela do BD associado ao *app*, e passar a gerar o arquivo de projeto após todas as informações necessárias serem fornecidas (e o botão adequado ser pressionado).

Outra possibilidade seria, apenas alterar o fluxo de execução, para que não seja mais necessário armazenar o arquivo de ontologia quando se adiciona novos criadores, mas somente quando o projeto, como um todo, é salvo.

4.3. Possíveis Melhorias

Aqui serão discutidas as possíveis melhorias que podem ser desenvolvidas.

São consideradas melhorias: novas adições as funcionalidades já implementadas, ou a elaboração de novos componentes e/ou recursos, que expandam ou melhorem a operação da ferramenta.

4.3.1. Documentação Básica

Como já discutido, o projeto do SysLODFlow não possui documentação além do relatório de Trabalho de Conclusão de Curso (e artigo relacionado) que deu origem a aplicação (MORAIS; MORAIS, 2016). Tal fato vai de encontro com as boas práticas de ES, as quais ditam que a falta de documentação leva a um aumento considerável da carga de trabalho de desenvolvedores que desejem trabalhar para expandir o *software* e leva a soluções que normalmente ficam abaixo do ideal (MALL, 2014).

A falta de arquivos de documentação - que expliquem processos básicos - também cria uma barreira para os novos operadores da ferramenta, que acabam por ter de descobrir de maneira exploratória como realizar as tarefas que desejam. Essa situação vai de encontro ao fundamento que deu origem ao SysLODFlow, que seria amenizar a carga imposta a quem deseja publicar *Linked Data*.

O que se propõe como primeiro passo para a solução do problema, é a criação de um documento *README*[16], devidamente formatado com a linguagem *markdown*[17], que no mínimo descreva o processo de instalação e execução das funcionalidades primárias do aplicativo. Caso seja possível, dentro das limitações de

[16] O documento *README* é uma forma básica de documentação, presente na maioria das ferramentas distribuídas em repositórios Git (guides.github.com/features/wikis/).

[17] commonmark.org/

tempo deste trabalho, as funcionalidades secundárias e desenvolvidas também devem ser descritas.

4.3.2. Incluir Padrão de Entrada JSON

Durante muitos anos o formato preferencial para transmissão de dados na *internet* foi o XML, contudo os problemas apresentados, o envelhecimento do padrão e o surgimento de novos formatos, fez com que o JSON fosse adotado como modelo padrão para troca de informações na rede. O modelo é descrito como um formato leve de intercâmbio de dados, que é de fácil compreensão para humanos, tanto em sua leitura quanto escrita, ao mesmo tempo que máquinas conseguem realizar fácilmente o *parsing* das informações contidas no arquivo (ECMA, 2013).

De posse dessas informações, e do conhecimento de que cada vez mais o JSON ganha espaço sobre os seus concorrentes diretos (CSV e XML), seria interessante que se adaptasse o SysLODFlow para aceitar o *input* dados formatados nesse padrão, além do suporte ao formato CSV já implementado.

A proposição descrita acima é apoiada pelos criadores originais da ferramenta, em sua seção de “Trabalhos Futuros” (MORAIS; MORAIS, 2016), o que dá um maior respaldo a idéia.

4.3.3. Adicionar Recursos Para Cancelar Operações

Múltiplas seções da interface não apresentam um modo de cancelar o procedimento sendo realizado, ou de retornar a página que levou o operador até aquela região do *software*.

Dessa maneira é conveniente adicionar algo como botões nas páginas relevantes, para que o fluxo de execução, bem como a experiência de uso, sejam aprimorados. Se as interfaces já apresentarem botões de cancelamento, averiguar a necessidade de se destacar esses com cores adequadas, ou melhor posicionamento na tela.

Também é preciso ressaltar, a necessidade dos botões, ou outro tipo de solução escolhida, não apresentarem os comportamentos adversos elencados no item 4.2.2, da subseção anterior deste documento.

4.3.4. Criar Testes Automatizados

Ao se criar *software* é necessário validar o código desenvolvido, para que seja assegurado que o que se pretendia seja realizado, bem como a qualidade do produto. Mais importante do que isso, é se assegurar que algo fora do esperado não ocorra em decorrência do que foi desenvolvido. A maior vantagem de se realizar testes é essa, a prevenção, descoberta e correção de erros e *bugs* no sistema.

Contudo, testar *software* é uma tarefa árdua e repetitiva e que, caso não seja feita de maneira padronizada, pode vir a deixar erros passarem em certas partes do código, ou a cada atualização nova. Para que isso não ocorra, existem diversos

tipos diferentes de testes automatizados que podem ser aplicados, os mais conhecidos sendo: testes unitários e testes de aceitação.

Certos autores, como Silveira Neto et al. (2011), definem os testes unitários como sendo as verificações das menores partes possíveis de um aplicativo, como funções, interfaces e classes.

Os autores também determinem testes de aceitação como sendo aqueles que, normalmente, são realizados pelos clientes e validam as operações através da interface do sistema, ou seja: são testes de uso real do sistema por usuários. Contudo estes também podem ser escritos por programadores e automatizados com a ajuda de algumas dispositivos.

As maiores vantagens destes tipos de checagem são o seu alto custo-benefício e a possibilidade de serem escritos pelos mesmos indivíduos que criaram o código. Dessa maneira é sugerido que se criem casos de teste para o SysLODFlow, já que seu escopo é abrangente e diversos programadores já se viram envolvidos com seu desenvolvimento. As verificações serviriam para que em adições futuras novos contribuidores possam ter um alicerce para validar suas soluções, de maneira que não necessitem mergulhar tão a fundo em todas as partes do código fonte.

Sugere-se começar o desenvolvimento das avaliações automáticas pelas duas variedades citadas, tanto por sua facilidade de elaboração, quanto pelo alto fator de melhoria, padronização e estabilidade do código de aplicações onde são aplicados.

4.3.5. Interfaces Para Geração ou Edição de Arquivos de Configuração

O funcionamento do SysLODFlow depende de vários artefatos de *software* que oferecem apoio a execução de seus *workflows*, como o LIMES[18] e o Sparqlify[19]. A configuração desses programas se dá através de arquivos, que devem ser criados manualmente pelos usuários. Novamente, essa situação aumenta o esforço necessário por parte do utilizador, e engessa o processo de reuso de *workflows*, pois não permite que se altere os arquivos de maneira fácil, para que sejam utilizados em outros projetos.

Aconselha-se criar interfaces que permitam a criação automática desses arquivos, de maneira o mais simples possível. Além disso é necessário que configurações já existentes possam ser carregadas e modificadas, em um processo simples e rápido.

Uma vez mais, a idéia apresentada é discutida no trabalho original na seção de “Trabalhos Futuros”, dando maior peso a necessidade de implementação do recurso.

[18] aksw.org/Projects/LIMES.html
[19] aksw.org/Projects/Sparqlify.html

4.4. Desenvolvimento das Melhorias e Correções do Projeto

Nesta seção serão discutidas as correções e melhorias selecionadas a partir das elencadas no segmento anterior deste trabalho, para desenvolvimento e adição ao projeto SysLODFlow. Também serão apresentadas as razões e raciocínio por trás das decisões tomadas.

Por fim, são expostas as soluções desenvolvidas e os desafios e decisões que tomou-se quanto a escrita do código.

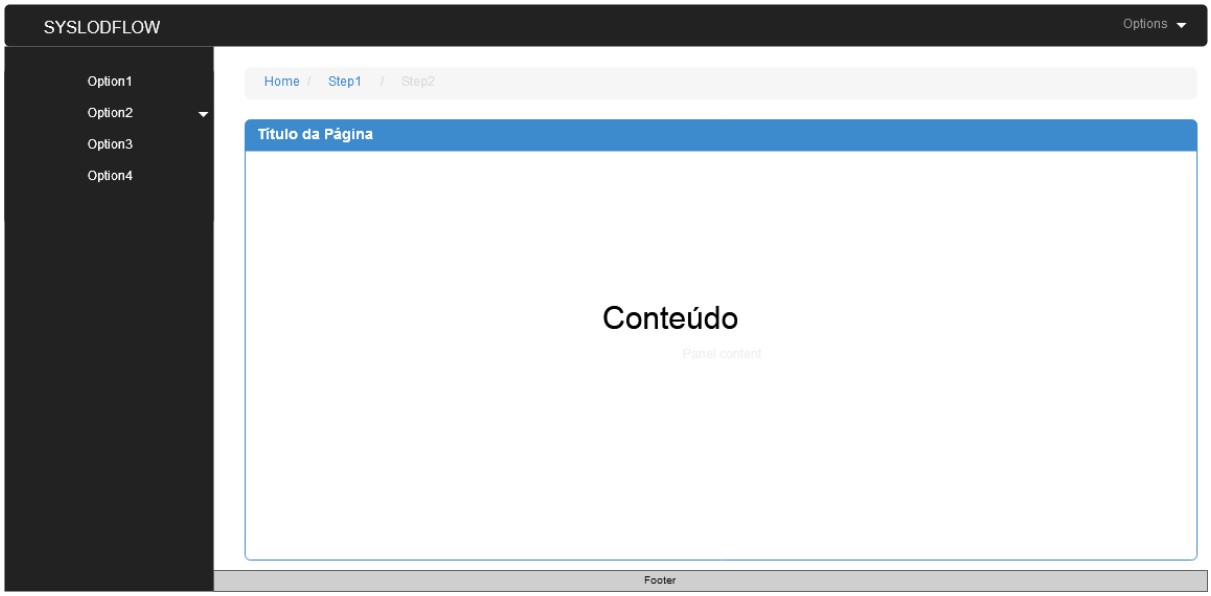
4.4.1. Renovação do Layout Geral das Páginas do Sistema

Como primeiro passo para o *update* no layout das páginas do aplicativo, foram atualizados para sua versão mais recente, os *frameworks* e bibliotecas utilizados na elaboração da GUI do sistema. As tecnologias atualizadas foram: *Bootstrap*, *Primefaces*[20], *DataTables*[21], *JQuery*[22] e *SB Admin*[23]. É importante ressaltar que todas as atualizações seguiram os devidos guias, quando esses estavam disponíveis através de fontes oficiais.

De posse de todos os recursos atualizados passou-se a reavaliar a interface gráfica do sistema, para que se pudesse desenvolver soluções que atendessem a todos os defeitos elencados anteriormente, bem como seguissem o mesmo padrão de design. Criou-se então o *mockup* da figura 15 (seguindo conceitos de *User Experience* discutidos previamente), como guia para criação e modificação de páginas do SysLODFlow.

[20] www.primefaces.org/
[21] datatables.net/
[22] jquery.com/
[23] startbootstrap.com/template-overviews/sb-admin/

Figura 17: Mockup ilustrando o layout padronizado das páginas do sistema.



Fonte: Autor.

O *mockup* apresentado conta com alguns componentes fixos, são eles:

- *Navbar* superior com nome do sistema e botão *dropdown*;
- *Navbar* lateral contendo o menu de navegação;
- *Footer* com informações do projeto;
- Área para apresentação do conteúdo.

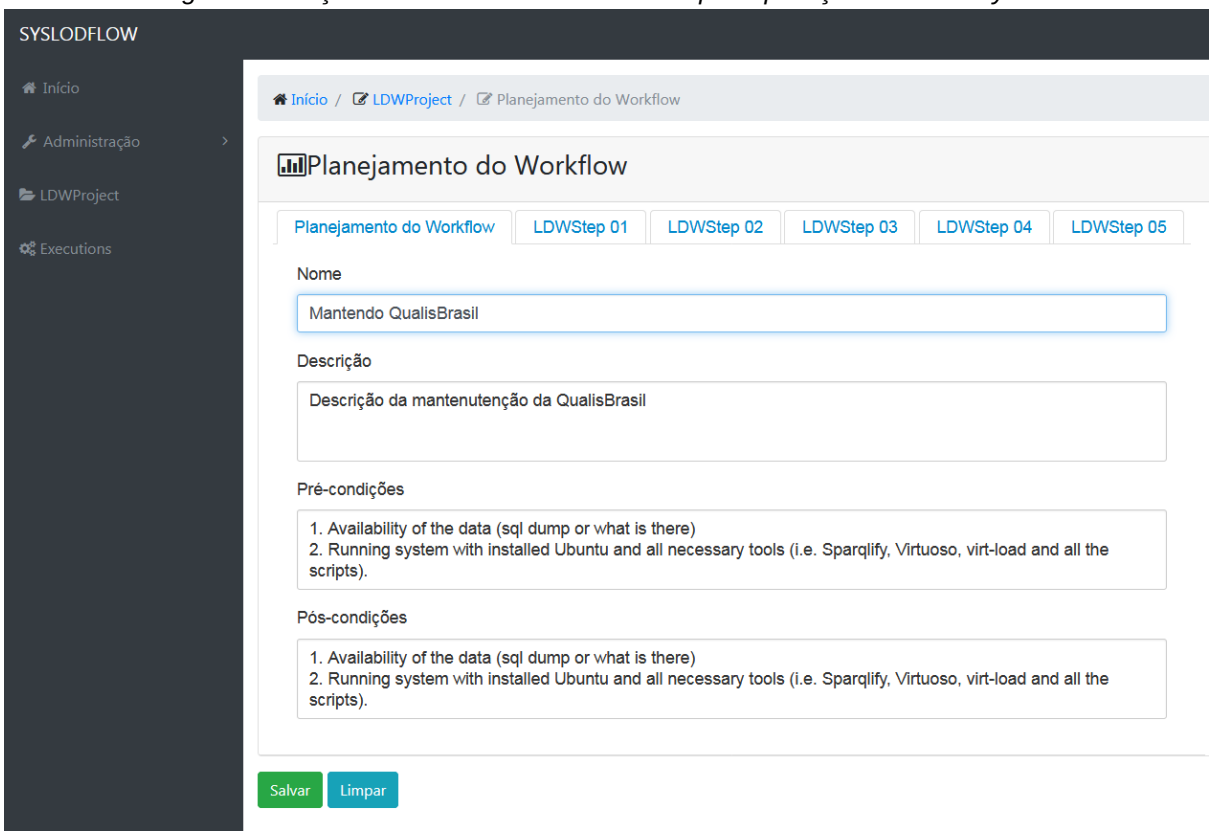
O útlimo componente observado é a navegação através de *breadcrumbs*[24], a qual não é fixa, aparecendo apenas quando uma profundidade de duas camadas é atingida pelo usuário, como recomendado no guia de estilo da IEEE (2017).

Com esse recurso servindo como guia, foi possível alcançar uma identidade visual para todas as páginas, o que fez com que a utilização do sistema fosse mais fluida e intuitiva. No futuro, quando novas páginas necessitarem ser desenvolvidas, o *mockup* deverá ser utilizado para orientar o processo.

Ademais com a aplicação correta do *Bootstrap,* o *layout* passou a ser responsivo de maneira que o sistema agora também será exibido consistentemente em dispositivos *mobile* e nos três principais *browsers* da atualidade. Além disso a aplicação das classes CSS atualizadas fez com que os componentes visuais não estejam mais passando dos limites de altura e largura de seus contêineres, assim como padronizou o alinhamento e largura de todos elementos. Os resultados obtidos podem ser observados na figura 18.

[24] A navegação por *breadcrumbs* fornece *links* para cada página que o usuário navegou e mostra a localização atual do usuário dentro do site (www.w3schools.com/howto/howto_css_breadcrumbs.asp).

Figura 18: Seção de cadastro de Workflows após aplicação do novo layout.



Fonte: Autor.

Durante este processo todo, os arquivos XHTML foram limpos e indentados corretamente, prezando pela legibilidade do código. Isso foi realizado para que as tarefas descritas aqui pudessem ser feitas de maneira mais fácil e correta, bem como para preparar esses arquivos para manuseio mais fácil por parte de futuros contribuidores do projeto.

4.4.2. Adição de Botões Para Cancelamento de Operações

Como se decidiu padronizar melhor a GUI do sistema esta alteração acabou por ser reavaliada. Considerando melhor as opções e conceitos de UX, decidiu-se por não adicionar novos botões às páginas, pois acredita-se que o benefício à navegação seja mínimo ao se fazer uso dessa alternativa.

Como já mencionado anteriormente foi preferida a navegação por *breadcrumbs* em todas as páginas do sistema. Permitindo que se possa retornar à página de origem, ou ao *index*, assim dando uma opção intuitiva para se interromper a execução de uma atividade, a qual tem um impacto visual e prático maior do que um simples botão.

4.4.3. Correção de Botões com Erros e *Bugs* Gerados por Eles

Como já dito antes, os erros relacionados aos botões do *app* eram diversos e não correlacionados. Foi decidido tratar cada um dos problemas individualmente nesse segmento da obra. A seguir estão descritos os *bugs* e procedimentos de conserto adotados.

4.4.3.1. Botão Salvar da Tela de Cadastro de Autores

O problema com este elemento específico era que o formulário, de onde a janela *modal* que continha o botão era originada, era submetido (o que estava ativando as verificações de preenchimento de campo observadas). Isso não deveria ocorrer, pois este botão pertence a outro formulário.

O comportamento anômalo era causado pela colocação equivocada do XHTML da janela *modal* dentro das *tag*s do formulário principal.

A correção para este erro foi simple: o código XHTML, referente a janela do cadastro de autores, foi movido para uma localidade apropriada no documento e envolto em *tags* de formulário. O campo “Nome” também foi marcado para preenchimento obrigatório e ganhou mensagens de erro personalizadas.

A seção de cadastro de autores ao projeto passou então a apresentar um comportamento consistente, correto e com prevenção de erros e falhas.

4.4.3.2. Botão Voltar da Tela de Execução de *Workflow*

O defeito apresentado aqui é similar ao tratado no item 4.4.3.1: o código do botão encontrava-se dentro das *tags* de formulário, o que causava a submissão do mesmo quando o botão era pressionado.

Após a decisão de se padronizar a GUI e utilizar-se de *breadcrumbs* para navegação, este botão passou a ser obsoleto. Logo a decisão satisfatória passou a ser a remoção completa do botão, o que foi prontamente executado.

4.4.3.3. Botões *Download* e Excluir

Neste caso, todos os botões relacionados a *download* e exclusão de arquivos não tinham suas funcionalidades implementadas.

Como isto não é um erro propriamente dito, a única correção disponível é a implementação das funcionalidades planejadas. Para tal o *framework Primefaces* conta com recursos para auxiliar na implementação destes cenários.

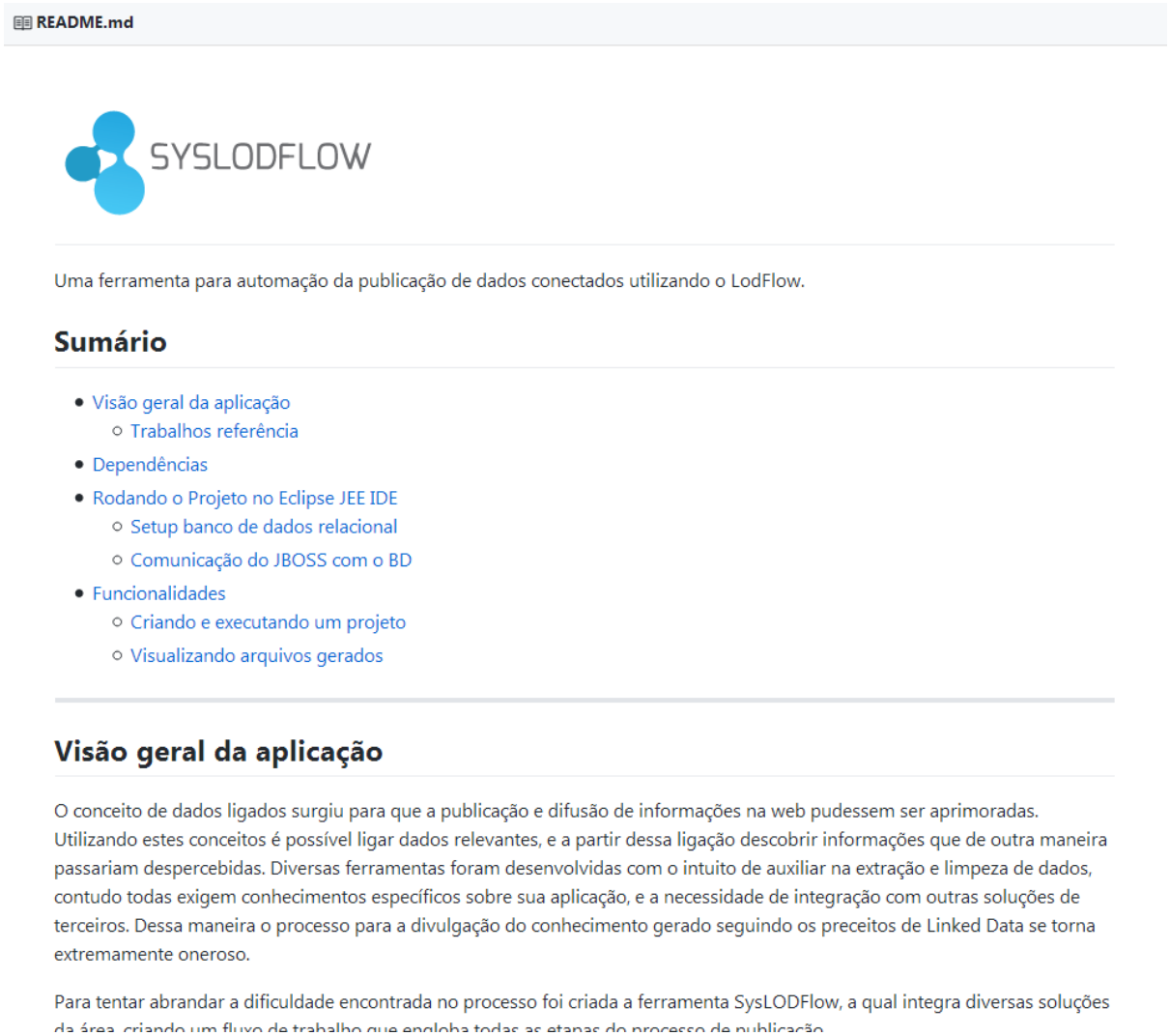
Dito isso, os botões tiveram suas ações desenvolvidas, de maneira a assegurar que o usuário tivesse todas as informações e mensagens de confirmação necessárias a cada tarefa. Os mecanismos disponíveis através do *Primefaces* foram adotados a fim de se assegurar um funcionamento melhor e livre de erros.

4.4.4. Documentação Básica

Para sanar o problema da falta de documentação, foi desenvolvido um arquivo *README* no formato *markdown*. Como o projeto está sendo publicado no GitHub, a sintaxe utilizada foi a ditada pelo guia do mesmo, para que a visualização do arquivo seja a melhor possível. O documento original pode ser observado no apêndice A.

Um fato interessante que ocorre é a interpretação do *README* pelo GitHub, o qual transforma texto formatado em arquivo interpretado e com diversos recursos visuais. Esse mecanismo acaba por gerar uma visualização que é esteticamente agradável e de fácil assimilação pelo maior número de usuários. O resultado desse processo pode ser constatado na figura 19, a qual mostra como o GitHub exibe o texto do apêndice.

Figura 19: Arquivo README visualizado no repositório GitHub.



Fonte: Autor.

Durante a elaboração da documentação tentou-se criar um guia que fosse claro, breve e seguisse boas recomendações para elaboração deste tipo de documento (THOMPSON, 2016) (SINGERS, 2014).

O que se desejava deste arquivo era que ele explicasse os principais métodos de uso do aplicativo, o método de instalação e pontos em que houve dificuldade durante a instalação e operação do SysLODFlow, ou que são necessários para a compreensão do funcionamento do *app*, mas que não são óbvios ou descritos de maneira simples. Outros temas e recursos menos importantes (mas que ainda assim ajudam a quem deseja entender e trabalhar com o programa) foram adicionados, como os trabalhos que levaram ao desenvolvimento do SySLODFlow.

Assim os assuntos abordados no *README* acabaram por ser:

- Visão geral da aplicação;
- Trabalhos referência;
- Dependências;

- Árvore de diretórios;
- Rodando o projeto no Eclipse JEE IDE;
- Instalando dependências;
- Setup banco de dados relacional;
- Comunicação do JBOSS com o BD;
- Utilização;
- Criando e executando um projeto;
- Criando execuções de fluxos e visualizando arquivos gerados;
- Créditos.

Dessa maneira a documentação proporciona rapidamente aos usuários as noções necessárias para que eles entendam a idéia geral do projeto, tenham uma idéia geral do seu funcionamento e instalem o *software* através de instruções simplificadas. Isso é extremamente importante para que contribuidores novos se aclimatem mais rapidamente ao ambiente, pois terão orientações para seguir e fontes de informação caso desejem entender melhor as decisões de design do programa.

Deste modo criou-se um documento que supre as necessidades básicas de documentação, aliviando a carga de pesquisa e aprendizado necessária para se colaborar para o SysLODFlow. Todavia isso serve apenas como primeiro passo, não sendo o suficiente para substituir uma documentação mais extensa e aprofundada.

4.4.5. Correção das propriedades da classe Usuario

Apesar da decisão mais correta seja a adequação do sistema, para trabalhar com o cadastro de usuários utilizando o campo de e-mail, decidiu-se por retirar a propriedade (e todas as referências a ela) dos arquivos do *app*.

Chegou-se a essa resolução pelo fato de que, embora seja necessário implementar um cadastro de operadores mais robusto, em seu seu presente estado o SysLODFlow viria a ganhar muito pouco com isso. Outras necessidades mais urgentes precisam ser endereçadas antes que se almeje expandir estas funcionalidades.

4.4.6. Correção das Exceções no Fluxo de Operações de Criação de Arquivos de Projeto

Em primeiro momento procurou-se transferir as informações de autores para uma tabela do banco de dados MySQL[25] (o qual já é uma dependência do projeto). Entretanto, o esforço exigido para isso seria significativo, pois seria necessário alterar a forma como o *app* estava recuperando e escrevendo essas informações na ontologia - um ponto já tangenciado por correções discutidas anteriormente e que,

[25] www.mysql.com/

caso implementado, alteraria a lógica de relacionamento da ferramenta com a ontologia.

Dessa forma, foi decidido implantar, além dos aprimoramentos discutidos na subseção 4.4.3.1 deste documento, a alteração do fluxo de execução das operações, para que os dados sobre o autor sejam salvos juntamente com o resto do projeto. Assim situações como a descrita, onde um projeto nulo foi salvo de maneira incorreta, são evitadas de forma consistente e permanente.

4.4.7. Reestruturação do Processo de Criação e Execução de LDWProjects e Arquivos de Ontologia

Como discutido no item 4.2.3 os principais problemas avaliados são referentes a ontologia e as implementações relacionadas a ela, configurando um problema grave que afeta quase todas as partes do sistema. Apesar disso se decidiu começar o desenvolvimento solucionando os problemas menores descritos, pois esses provêm apetrechos e recursos necessários a execução de *workflows* e são um ponto de partida ideal para uma questão tão ampla como a proposta aqui.

Sobre a execução de queries SQLs em bancos relacionais, foi optado por não abordar o tópico em primeiro momento, visto que o acesso a cada banco (bem como a versão da linguagem SQL suportada) é diferente, o que faz com que esta seja uma tarefa de alta complexidade e que precisa ser abordada após um exemplo de execução de *workflows* estiver funcionando.

O que se fez foi começar o processo pelo desenvolvimento de métodos de *upload* para arquivos de *dataset* CSV. Isso foi feito tendo em vista o fato de que o passo 6 da figura 11 cita o processo de extração de dados e subsequente transformação desses para o formato, para que sejam trabalhados pelo Sparqlify, além do que nos projetos de exemplo o primeiro passo - que seria de extração de dados e conversão para o CSV - foi pulado já que um arquivo adequado ao processamento do Sparqlify é fornecido (RAUTENBERG, 2015). Foi adicionada uma opção durante a criação do projeto para que o usuário escolha entre o *upload* de um arquivo CSV (o que implica na necessidade de se ignorar a parte de coleta de dados de um banco de dados) e a execução de uma *query* SQL em um BD relacional - funcionalidade que ainda não está operante.

O próximo passo foi corrigir os *shell scripts* gerados quando se produz um novo *workflow*. Os comandos colocados nos *scripts* continham erros de sintaxe e, principalmente, não forneciam endereços consistentes entre si. Isso acarretava em alguns *scripts* se referirem a arquivos gerados pela execução anterior de outro *script*, mas com um caminho absoluto incorreto, resultando em uma exceção quando os arquivos necessários não eram encontrados. Para solucionar esses impasses foi decidida em uma estrutura de diretórios que separe os arquivos gerados de forma coerente, significando que arquivos com diferentes extensões ficam segregados, e os comandos em si foram alterados para que executem corretamente referenciando os caminhos das pastas de projeto corretas.

Quanto as ferramentas não disponibilizadas a correção é simples: os arquivos executáveis JAR foram adicionadas aos arquivos do projeto, assim possibilitando seu uso pelo SysLODFlow.

Finalmente, após ter todos os problemas menores solucionados, pode-se começar a reparar as falhas relativas a ontologia LDWPO. A primeira modificação necessária seria a substituição do modelo de ontologia utilizado atualmente, por um arquivo de ontologia LDWPO “virgem”, e subsequente alteração dos métodos de edição e leitura para que os arquivos LDWPO gerados sejam compatíveis com o LODFlowEngine. Ao se estudar os arquivos de exemplo da Qualis Brasil notou-se que eles foram gerados utilizando a biblioteca OWL API (THE UNIVERSITY OF MANCHESTER, 2017), a qual também pode ser aplicada aqui para solucionar o problema de compatibilidade encontrado.

4.4.8. Correção de Arquivos XHTML Com Erros de Sintaxe

Enquanto se realizavam todas as tarefas descritas até este momento, quaisqueres erros de sintaxe em páginas XHTML foram devidamente corrigidos, para que essas fossem exibidas de maneira apropriada.

5. CONCLUSÃO E TRABALHOS FUTUROS

O que se propunha neste trabalho era o aprimoramento do SysLODFlow nas áreas onde se julgava existirem deficiências, ou onde os processos implementados podiam ter seu funcionamento expandido e/ou melhorado. Todavia o que se encontrou - após um estudo aprofundado e demorado do código da aplicação - foi uma ferramenta que continha diversos erros, *bugs* e equívocos em seu código, além de não atender a todos os requisitos e ideais delineados na obra que deu origem ao projeto. Após a constatação do estado inacabado do *webapp* foi decidido que era necessário alterar o foco deste projeto para algo de natureza mais corretiva ao invés de evolutiva, como foi originalmente pensado. O que se fez então foi um trabalho de mapeamento das áreas errôneas e deficientes do código, a correção dos defeitos mais urgentes e o desenvolvimento de recursos para que os conceitos por trás do sistema, e sua operação, fossem listados e sintetizados para compreensão de futuros contribuintes (seguindo a idéia de que este é um *software* livre e aberto para uso e cooperações). A tabela 1 fornece a visão geral sobre quais propostas elencadas foram adotadas ou não implementadas até o momento de conclusão deste trabalho.

Tabela 1: Visão geral da situação das soluções propostas.

Erro e/ou Melhoria Relacionada	Situação das Soluções Propostas
Layout geral das páginas do sistema	Implementadas
Botões apresentando diversos comportamentos errôneos ou não funcionando	Implementadas
Processo de criação e execução de LDWProjects e arquivos de ontologia	Parcialmente Implementadas
Arquivos XHTML com erros de sintaxe	Implementadas
Strings com erros	Implementadas
Exceções não tratadas	Implementadas
Classe usuário com propriedades conflitantes	Implementadas
Projetos novos não separados em pastas distintas	Não implementadas
Documentos de projeto sendo criados mesmo que o processo não tenha sido completado corretamente	Implementadas
Documentação básica	Implementadas
Incluir padrão de entrada JSON	Não implementadas
Adicionar recursos para cancelar operações	Solução alternativa implementada
Criar testes automatizados	Não implementadas
Interfaces Para Geração ou Edição de Arquivos de Configuração	Não implementadas

Fonte: Autor.

Nas subseções que virão são exploradas as conclusões finais relacionadas ao que foi realizado ao longo deste trabalho e sugestões de novas empreitadas que podem ser realizadas a partir do que já foi produzido ou vislumbrado durante a implementação das soluções descritas.

5.1. Conclusão

Chegando ao final dos trabalhos realizados foram geradas centenas de novas linhas de código, diversas dependências atualizadas e dezenas de *commits* submetidos ao repositório GitHub que hospeda o projeto, que acabaram por fazer com que o SysLODFlow se tornasse uma ferramenta muito mais completa, funcional e que atende aos requisitos funcionais estabelecidos e descritos em sua concepção inicial. Todas as alterações, e código fonte do aplicativo, podem ser encontradas no endereço *github.com/bruno-edo/syslodflow_01*, onde o projeto estará perpétuamente disponível. Ademais, seguindo o espírito do *software* livre, novos colaboradores e contribuições serão prontamente aceitas após revisão das

alterações submetidas, para a certificação de que estão de acordo com as regras de negócio estabelecidas.

5.2. Trabalhos Futuros

A partir de todo trabalho de pesquisa e Engenharia de *Software* desenvolvidos aqui, fica claro o quanto ainda pode-se fazer para evoluir e aprimorar o funcionamento do SysLODFlow.

De posse do conhecimento gerado neste TCC as possibilidades de desenvolvimento são amplas, mas algumas delas acabam por ser mais urgentes que outras - já que não se pode tocar em todos os pontos que se desejava durante a etapa de codificação deste trabalho. Com isso em mente é recomendado que no futuro sejam desenvolvidos os seguintes tópicos:

- Criação de testes automatizados que aumentem a confiabilidade de novas submissões (*commits*) ao código do SysLODFLow, bem como auxilie os desenvolvedores;
- Elaboração de interface para criação e edição de arquivos de configuração para as ferramentas de apoio Sparqlify e LIMES;
- Criação de interface para a reutilização de propriedades de projetos antigos no cadastro de novos projetos e *workflows*;
- Adição de suporte a novas ferramentas de apoio que realizem tarefas semelhantes ao LIMES e Sparqlify;
- Expansão e melhoria da GUI do *app*, seguindo os conceitos de *User Experience*, boas práticas, usabilidade e acessibilidade;
- Expansão e melhoria do suporte a múltiplos usuários cadastrados e utilizando a aplicação, para que se passe a ter um ambiente com diversos usuários criando e compartilhando projetos próprios (com a devida separação de arquivos e tratamento de exceções);
- Documentação do projeto mais aprofundada, abrangente e técnica. Cobrindo aspectos como funcionamento dos métodos e ferramentas de apoio;
- Tratamento de quaisquer erros, exceções ou melhorias discutidos anteriormente neste trabalho, mas que não puderam ser desenvolvidos devido a constrições de tempo;
- Desenvolvimento novos modelos de *workflows* que podem ser adicionados a projetos criados;
- Melhoria e expansão do SysLODFlow no que tange a aspectos fora do escopo inicial da aplicação;
- Melhora do código desenvolvido a fim de aumentar a performance, legibilidade e confiabilidade do mesmo.

REFERÊNCIAS

[1] BERNERS-LEE, Tim. **Linked Data:** design issues. 2006. Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html>. Acesso em: 07 nov. 2016.

[2] AUER, Sören et al. Introduction to Linked Data and Its Lifecycle on the Web. **Reasoning Web. Semantic Technologies For Intelligent Data Access,** [s.l.], p.1-90, 2013. Springer Science + Business Media. http://dx.doi.org/10.1007/978-3-642-39784-4_1.

[3] RAUTENBERG, Sandro et al. LODFlow: a workflow management system for linked data processing. **Proceedings Of The 11th International Conference On Semantic Systems - Semantics '15,** [s.l.], p.137-144, set. 2015. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/2814864.2814882. Disponível em: <http://eis.iai.uni-bonn.de/blog/wp-content/uploads/2015/08/SEMANTiCS_2015_Research_Track_submission_96.pdf>. Acesso em: 2 nov. 2016.

[4] MORAIS, Jean Carlos de; MORAIS, Jhonatan Carlos de. **SysLodFlow:** Uma Ferramenta de Apoio a Automação Da Publicação e Manutenção de Linked Data Utilizando o LODFlow. 2016. 77 f. TCC (Graduação) - Curso de Sistemas de Informação, Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2016.

[5] IEEE Computer Society (Org.). **Friedrich L. Bauer:** 1988 Computer Pioneer Award. Disponível em: <https://www.computer.org/web/awards/pioneer-friedrich-bauer>. Acesso em: 10 jun. 2017.

[6] BOEHM, Barry W.. Seven basic principles of software engineering. **Journal Of Systems And Software,** [s.l.], v. 3, n. 1, p.3-24, mar. 1983. Elsevier BV. http://dx.doi.org/10.1016/0164-1212(83)90003-1.

[7] IBM (Estados Unidos). **Application Programming on z/OS.** [s.l.]: Z/os Basic Skills Information Center, 2006. 90 p. Disponível em: <https://www.ibm.com/support/knowledgecenter/zosbasics/com.ibm.zos.zappldev/zappldev_book.pdf?view=kc>. Acesso em: 06 dez. 2017.

[8] VAN DEURSEN, Arie; BURD, Elizabeth. Software reverse engineering. **Journal Of Systems And Software,** [s.l.], v. 77, n. 3, p.209-211, set. 2005. Elsevier BV. http://dx.doi.org/10.1016/j.jss.2004.03.031. Disponível em: <https://pdfs.semanticscholar.org/a5a9/4454daa877d1ebbac52b9c8ba9a0f94d786b.pdf>. Acesso em: 05 set. 2017.

[9] CIPRESSO, Teodoro; STAMP, Mark. Software Reverse Engineering. In: STAVROULAKIS, Peter; STAMP, Mark. **Handbook Of Information And Communication Security.** [s.l]: Springer Berlin Heidelberg, 2010. Cap. 31. p. 659-696.

[10] BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. "The Semantic Web" by, Scientific American. (2001). Disponível em <http://www.disciplineoforganizing.org/wp-content/uploads/2013/01/SemanticWeb.pdf>. Acesso em: 7 jun. 2017.

[11] W3C (Brasil). **Web Semântica.** Disponível em: <http://www.w3c.br/Padroes/WebSemantica>. Acesso em: 20 maio 2017.

[12] HEATH, Tom. Linked Data Organization. **Linked Data - Connect Distributed Data across the Web:** Frequently Asked Questions (FAQs). 2009. Disponível em: <http://linkeddata.org/faq>. Acesso em: 05 jun. 2017.

[13] BERNERS-LEE, Tim. **The Linked Open Data Movement.** [s.i]: W3c, 2008. 22 slides, color. Disponível em: <https://www.w3.org/2008/Talks/0617-lod-tbl>. Acesso em: 05 jun. 2017.

[14] BERNERS-LEE, Tim et al. Tabulator: Exploring and Analyzing linked data on the Semantic Web. **Proceedings Of The 3rd International Semantic Web User Interaction Workshop: SWUI,** Massachussets, v. 1, n. 1, p.1-16, jan. 2006. Disponível em: <http://hmslydia.com/my_publications/BernersLeeTabulator2006.pdf>. Acesso em: 05 jun. 2017.

[15] GRUBER, Thomas. A Translation Approach to Portable Ontology Specifications. **Knowledge Acquisition,** Stanford, v. 2, n. 5, p.190-220, abr. 1993. Disponível em: <http://tomgruber.org/writing/ontolingua-kaj-1993.pdf>. Acesso em: 05 jun. 2017.

[16] GUARINO, Nicola. Formal Ontology and Information Systems. **Proceedings Of Fois'98,** Trento, Itália, v. 8, n. 6, p.3-15, jun. 1998. Disponível em: <http://www.loa.istc.cnr.it/old/Papers/FOIS98.pdf>. Acesso em: 07 jun. 2017.

[17] GRUBER, Tom. Ontology. **Encyclopedia Of Database Systems,** [s.i], v. 1, n. 1, p.1-4, jan. 2009. Disponível em: <http://tomgruber.org/writing/ontology-definition-2007.htm>. Acesso em: 06 jun. 2017.

[18] MAEDCHE, A.; STAAB, S.. Ontology learning for the Semantic Web. **Ieee Intelligent Systems,** [s.l.], v. 16, n. 2, p.72-79, mar. 2001. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/5254.920602.

[19] FERRUCCI, David et al. Building Watson: An Overview of the DeepQA Project. **AI Magazine,** [s.l.], v. 31, n. 3, p.59-79, Fall 2010.

[20] W3C (Estados Unidos). **SPARQL 1.1 Query Language.** 2013. Disponível em: <https://www.w3.org/TR/sparql11-query/>. Acesso em: 03 nov. 2016.

[21] W3C (Estados Unidos). **RDF 1.1 Concepts and Abstract Syntax:** W3C Recommendation 25 February 2014. 2014. Disponível em: <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>. Acesso em: 26 maio 2017.

[22] HEATH, Tom; BIZER, Christian. Linked Data: Evolving the Web into a Global Data Space. **Synthesis Lectures On The Semantic Web: Theory and**

Technology, [s.l.], v. 1, n. 1, p.15-16, 9 fev. 2011. Morgan & Claypool Publishers LLC. http://dx.doi.org/10.2200/s00334ed1v01y201102wbe001.
[23] BIZER, Chris; CYGANIAK, Richard; HEATH, Tom. **How to Publish Linked Data on the Web.** 2007. Disponível em: <http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/>. Acesso em: 25 maio 2017.
[24] JANSSEN, Marijn; CHARALABIDIS, Yannis; ZUIDERWIJK, Anneke. Benefits, Adoption Barriers and Myths of Open Data and Open Government. **Information Systems Management,** [s.l.], v. 29, n. 4, p.258-268, set. 2012. Informa UK Limited. http://dx.doi.org/10.1080/10580530.2012.716740.
[25] MOLLOY, Jennifer C.. The Open Knowledge Foundation: Open Data Means Better Science. **Plos Biology,** [s.l.], v. 9, n. 12, p.1-1, 6 dez. 2011. Public Library of Science (PLoS). http://dx.doi.org/10.1371/journal.pbio.1001195.
[26] LIU, Xue; HEO, Jin; SHA, Lui. Modeling 3-Tiered Web Applications. **13th Ieee International Symposium On Modeling, Analysis, And Simulation Of Computer And Telecommunication Systems,** [s.l.], v. 1, n. 1, p.1-8, set. 2005. IEEE. http://dx.doi.org/10.1109/mascots.2005.40.
[27] RAUTENBERG, Sandro et al. Linked data workflow project ontology: uma ontologia de domínio para publicação e preservação de dados conectados. **Tendências da Pesquisa Brasileira em Ciência da Informação,** [s.i], v. 9, n. 2, p.1-17, set./dez. 2016. Semestral. Disponível em: <http://inseer.ibict.br/ancib/index.php/tpbci/article/viewArticle/252>. Acesso em: 12 out. 2017.
[28] ECLIPSE FOUNDATION. **Eclipse IDE.** Disponível em: <https://www.eclipse.org/>. Acesso em: 12 out. 2017.
[29] OTTO, Mark; THORNTON, Jacob. **Migrating to v4.** Disponível em: <https://v4-alpha.getbootstrap.com/migration/>. Acesso em: 30 set. 2017.
[30] KIM, Song-kyoo. Extensive Methodologies for Design of User Experience by Systematic Innovations. **International Journal Of Social Science And Humanity,** [s.l.], p.100-102, 2013. EJournal Publishing. http://dx.doi.org/10.7763/ijssh.2013.v3.204. Disponível em: <http://www.ijssh.org/papers/204-D00019.pdf>. Acesso em: 05 set. 2017.
[31] MALL, Rajib. Requirements Analysis and Specification. In: MALL, Rajib. **Fundamentals of Software Engineering.** 4. ed. Delhi: Phi Learning, 2014. Cap. 4. p. 68-69.
[32] RAUTENBERG, Sandro. **Linked Data Workflow Project Ontology.** [s.i]: Aksw, 2015. 54 p. Disponível em: <https://github.com/AKSW/ldwpo/blob/master/misc/technicalReport/LDWPO_technicalReport.pdf>. Acesso em: 20 out. 2017.
[33] ECMA INTERNATIONAL. **ECMA-404**: The JSON Data Interchange Format. 1 ed. Genebra: Ecma International, 2013. 14 p. Disponível em: <https://www.ecma-

international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Acesso em: 12 jun. 2017.
[34] SILVEIRA NETO, Paulo Anselmo da Mota et al. A systematic mapping study of software product lines testing. **Information And Software Technology,** [s.l.], v. 53, n. 5, p.407-423, maio 2011. Elsevier BV. http://dx.doi.org/10.1016/j.infsof.2010.12.003.
[35] IEEE. **Style Guide:** Navigation and Linking. 2017. Disponível em: <https://brand-experience.ieee.org/guidelines/digital/style-guide/navigation-linking/#breadcrumbnavigation>. Acesso em: 20 out. 2017.
[36] THOMPSON, Billie. **A template to make good README.md.** 2016. Disponível em: <https://gist.github.com/PurpleBooth/109311bb0361f32d87a2>. Acesso em: 21 out. 2017.
[37] SINGERS, Matias. **A curated list of awesome READMEs.** 2014. Disponível em: <https://github.com/matiassingers/awesome-readme>. Acesso em: 21 out. 2017.
[38] THE UNIVERSITY OF MANCHESTER (Reino Unido). **The OWL API.** Disponível em: <http://owlcs.github.io/owlapi/>. Acesso em: 21 out. 2017.

APÊNDICE A - Arquivo README.md


```
![SysLODFlow](https://github.com/bruno-edo/syslodflow_01/blob/master/syslodflow/src/main/webapp/imagens/syslodflow.png "SysLODFlow")
==================
Uma ferramenta para automação da publicação de dados conectados utilizando o LodFlow.

## Sumário

- [Visão geral da aplicação](#visão-geral-da-aplicação)
  - [Trabalhos referência](#trabalhos-referência)
- [Dependências](#dependências)
- [Árvore de diretórios](#arvore-de-diretorios)
- [Rodando o projeto no Eclipse JEE IDE](#rodando-o-projeto-no-eclipse-jee-ide)
  - [Instalando dependências](#instalando-dependencias)
  - [Setup banco de dados relacional](#setup-banco-de-dados-relacional)
  - [Comunicação do JBOSS com o BD](#comunicação-do-jboss-com-o-bd)
- [Utilização](#utilizacao)
  - [Criando e executando um projeto](#criando-e-executando-um-projeto)
  - [Criando execuções de fluxos e visualizando arquivos gerados](#criando-execucoes-de-fluxos-e-visualizando-arquivos-gerados)
- [Créditos](#creditos)


---

## Visão geral da aplicação

O conceito de dados ligados surgiu para que a publicação e difusão de informações na web pudessem ser aprimoradas. Utilizando estes conceitos é possível ligar dados relevantes, e a partir dessa ligação descobrir informações que de outra maneira passariam despercebidas. Diversas ferramentas foram desenvolvidas com o intuito de auxiliar na extração e limpeza de dados, contudo todas exigem conhecimentos específicos sobre sua aplicação, e a necessidade de integração com outras soluções de terceiros. Dessa maneira o processo para a divulgação do conhecimento gerado seguindo os preceitos de Linked Data se torna extremamente oneroso.
```

Para tentar abrandar a dificuldade encontrada no processo foi criada a ferramenta SysLODFlow, a qual integra diversas soluções da área, criando um fluxo de trabalho que engloba todas as etapas do processo de publicação.

O SysLODFlow faz uso das funcionalidades do [LODFlow](https://github.com/AKSW/LODFlow), mas fornece opções mais “user friendly” (e automatizadas) para se realizar o ciclo de publicação e manutenção de dados conectados. A ferramenta também foi desenvolvida de maneira a ser um aplicativo web, o que soluciona algumas restrições iniciais do LODFlow, e abrem as portas para funcionalidades que antes não eram possíveis em aplicações standalone, mas que trazem consigo algumas necessidades pontuais de design para web.

O aplicativo web utiliza o Java EE 7 como principal linguagem de desenvolvimento, além de alguns frameworks e bibliotecas, que podem ser vistos, bem como em quais camadas são aplicados, na imagem a seguir.

![Stack de tecnologias do SysLODFlow](https://github.com/bruno-edo/syslodflow_01/blob/master/images/syslodflow-layers.png "Stack de tecnologias do SysLODFlow")

Trabalhos referência

O SysLODFlow começou a ser desenvolvido através de Trabalhos de Conclusão de Curso (TCCs) da Universidade Federal de Santa Catarina. Abaixo estão listados os trabalhos relacionados a este projeto:

- [LDWPO - Linked Data Workflow Project Ontology - Sandro Rautenberg](https://github.com/AKSW/ldwpo/blob/master/misc/technicalReport/LDWPO_technicalReport.pdf)
- [SysLODFlow – Uma ferramenta de apoio a automação da publicação e manutenção de Linked Data utilizando o LODFlow - Jean Carlos De Morais e Jhonatan Carlos De Morais - UFSC](https://github.com/bruno-edo/syslodflow_01/blob/master/pdfs/RELAT%20RIO%20-%20SYSLODFLOW%20%20-%20Jean%20e%20Jhonatan%20Carlos%20de%20Morais.pdf)
- [SysLODFlow: Ampliando as funcionalidades na automação da publicação e manutenção de dados abertos conectados usando o LodFlow - Bruno Eduardo D'Angelo de Oliveira - UFSC]()

Dependências

Para rodar o SysLODFlow são necessárias algumas ferramentas de apoio, bem como algumas tecnologias Java:

1. [Java EE 7](https://www.java.com/en/)
2. [JBOSS 7.1.0 (Thunder)](http://www.jboss.org/)
3. [LIMES](http://aksw.org/Projects/LIMES.html)*
4. [Sparqlify](http://aksw.org/Projects/Sparqlify.html)
5. [Virtuoso](https://virtuoso.openlinksw.com/)
6. [MySQL](https://www.mysql.com/)
7. [LODFlow](https://github.com/AKSW/LODFlow)*

Bibliotecas e Frameworks:

1. [Primefaces 6.1](https://www.primefaces.org/)*
2. [Bootstrap 4](https://getbootstrap.com/)*
3. [SB Admin](https://startbootstrap.com/template-overviews/sb-admin/)*
4. [DataTables](https://datatables.net/)*

> * Arquivos já inclusos neste projeto.

Árvore de diretórios

```
syslodflow
└── scripts
└── src
        └── main
        └── java.br.ufsc.inf.syslodflow
        |   └── business
        |   └── crypto
        |   └── dao
        |   └── dto
        |   └── entity
        |   └── service
        |   └── util
```

```
|   └── web
|
└── resources
|   └── br.com.connectionltda
|   └── configuration
|   └── META-INF
|
└── webapp
└── cadastros
└── css
└── estrutura
└── font-awesome
└── imagens
└── js
└── management
└── ontology
└── WEB-INF
```

Rodando o projeto no Eclipse JEE IDE

Após clonar o repositório do projeto, importe-o como um projeto [Maven](https://maven.apache.org/). Dessa maneira o Eclipse se encarregará de criar o arquivo **.war** automáticamente. Este arquivo será deployado para o JBOSS, para que a aplicação seja rodada por ele.

Para criar uma instância do JBOSS pelo Eclipse é necessário baixar o plugin JBOSS Tools, para isso acesse este [link](https://marketplace.eclipse.org/content/jboss-tools), ou (ainda dentro do Eclipse) abra o Eclipse Marketplace e procure por JBOSS Tools.

Após a instalação do plugin, acesse **New -> Other -> Server** e selecione o **JBOSS AS 7.1.0**. Depois deste passo, uma aba de servidores poderá ser observada na barra inferior da IDE, selecione-a e clique com o botão direito do mouse na instância do JBOSS e selecione a opção **Add and Remove...**, em seguida adicione o arquivo **.war**, do SysLODFlow, aos projetos que devem ser publicados pelo servidor e clique em Finish.

Após os outros passos do setup serem completos, basta rodar o servidor e o SysLODFlow deve ser publicado automáticamente por ele.

Instalando dependências

Para rodar o SysLODFlow é necessário instalar o Sparqlify e o Virtuoso.

Uma imagem do Sparqlify pode ser baixada instalada deste
[link](https://github.com/AKSW/Sparqlify/downloads).

Para instalar o Virtuoso basta seguir os passos deste
[link](http://vos.openlinksw.com/owiki/wiki/VOS/VOSUbuntuNotes).

Setup banco de dados relacional

No BD relacional escolhido, crie um esquema relacional chamado **syslodflowds** e rode o [script SQL de configuração](https://github.com/bruno-edo/syslodflow_01/blob/master/syslodflow/scripts/syslodflow.sql). Dessa maneira um usuário administrador padrão será criado com o usuário e senha abaixo:

> **Usuário:** admin
>
> **Senha:** admin

Comunicação do JBOSS com o MySQL

É necessário fazer o download do driver JDBC que se pretende utilizar (o conector MySQL pode ser baixado neste [link](https://www.mysql.com/products/connector/). Há também uma cópia do conector na pasta **syslodflow/src/main/resources/META-INF** deste projeto) e disponbilizá-lo para o JBOSS. Para isso siga estes passos:

1. Copie o driver do MySQL para a pasta **/modules/com/mysql/main**, do local onde o JBOSS foi instalado. Provavelmente não existirá a pasta **/mysql/main**, portanto você deverá criá-la e efetuar a cópia.
2. Na mesma pasta crie o documento module.xml contendo o código abaixo:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<module xmlns="urn:jboss:module:1.0" name="com.mysql">
  <resources>
        <resource-root path="mysql-connector-java-5.1.44.jar"/>
```

```
  </resources>
  <dependencies>
        <module name="javax.api"/>
  </dependencies>
</module>
```

A pasta main deverá conter os seguintes arquivos:

- module.xml
- mysql-connector-java-5.1.44.jar

Para que seja possível que o SysLODFlow se comunique com a instância do banco de dados relacional instalada, se faz necessário criar um **Data Source** no arquivo **standalone.xml** do JBOSS. Dessa maneira, procure em seu sistema pelo arquivo **standalone.xml** e navegue para a seção de datasources, após localizá-la inclua o trecho de código abaixo:

```xml
<datasource jta="true" jndi-name="java:jboss/datasources/syslodflowDS" pool-name="syslodflowDS" enabled="true" use-java-context="true" use-ccm="true">
        <connection-url>jdbc:mysql://localhost:3306/syslodflowds</connection-url>
        <driver>com.mysql</driver>
        <transaction-isolation>TRANSACTION_READ_COMMITTED</transaction-isolation>
        <pool>
        <min-pool-size>5</min-pool-size>
        <max-pool-size>30</max-pool-size>
        <prefill>true</prefill>
        <use-strict-min>false</use-strict-min>
        <flush-strategy>FailingConnectionOnly</flush-strategy>
        </pool>
        <security>
        <user-name>root</user-name>
        <password>root</password>
        </security>
        <statement>
        <prepared-statement-cache-size>32</prepared-statement-cache-size>
        </statement>
</datasource>
```

```

Navegue até a seção de drivers e insira o trecho abaixo.

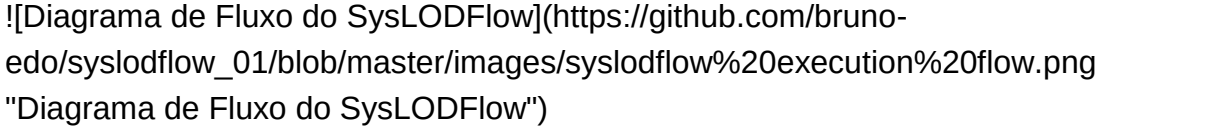
```xml
<driver name="com.mysql" module="com.mysql">
        <driver-class>com.mysql.jdbc.Driver</driver-class>
        <xa-datasource-
class>com.mysql.jdbc.jdbc2.optional.MysqlXADataSource</xa-datasource-class>
</driver>
```

> **Importante:** É preciso que sejam alterados os campos de URL da conexão ao banco, bem como o usuário e senha, para as informações de acesso configuradas para a instalação em seu sistema.
>
> O driver de comunicação também deve ser configurado de acordo com o sistema de banco de dados que se deseja utilizar. Neste caso estamos usando o MySQL como exemplo.

Após realizar essas etapas o projeto está pronto para ser lançado. Para isso, basta ativar o servidor JBOSS e o projeto será iniciado juntamente, no endereço IP e porta configurados.

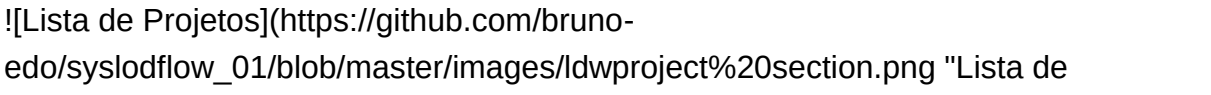
Utilização

O fluxo geral de operações do SysLODFlow pode ser visualizado no seguinte diagrama:



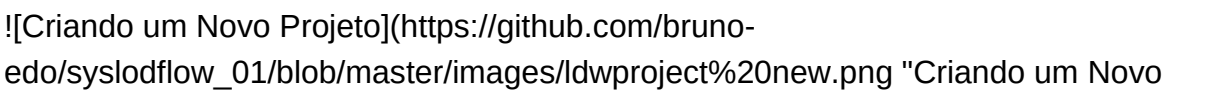
Criando e executando um projeto

Para visualizar os LDWProjects já cadastrados, acesse a seção LDWProject. Os projetos já cadastrados serão exibidos em uma lista.
```

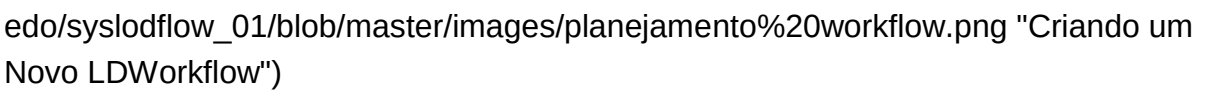


Para criar um novo projeto, clique no botão "Novo" e preencha os campos de informação. Após o preenchimento, clique em "Salvar" e o projeto deverá ser exibido na lista da figura anterior.

Para editar ou excluir um projeto, clique nos botões "Editar" e "Excluir", respectivamente.

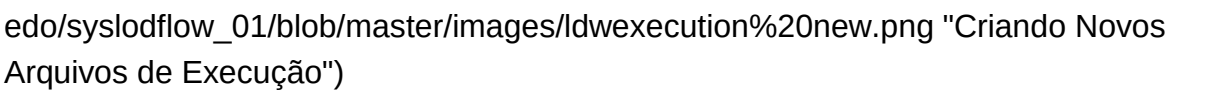


Para criar um workflow novo, clique no botão "Abrir" da coluna de LDWOrkflows da lista de projetos. Após o carregamento da página, preencha os campos com as informações relacionadas ao workflow que se deseja criar (como base de dados a ser acessada, arquivos de configuração do LIMES e do Sparqlify, entre outros).



Criando execuções de fluxos e visualizando arquivos gerados

Para criar novos arquivos de execução de fluxo, acesse a parte de LDWorkflowExecutions e clique no botão Novo. Insira as informaçãoes necessárias e clique em "Salvar" para armazenar os dados em disco.



Caso se deseje executar um fluxo, basta clicar no botão "Executar", e o fluxo deverá ser enviado ao LODFlow engine. Após o processo ser finalizado, os arquivos gerados poderão ser visualizados na página principal de LDWorkflowExecution, como mostrado abaixo.

![Lista de Arquivos Gerados](https://github.com/bruno-edo/syslodflow_01/blob/master/images/ldwexecution.png "Lista de Arquivos Gerados")

Créditos

[Bruno Eduardo D'Angelo de Oliveira](https://github.com/bruno-edo)

[Jean Carlos de Morais](https://github.com/jeanmorais/)

[Jhonatan Carlos de Morais]()

APÊNDICE B – Artigo

SysLODFlow - Ampliando as funcionalidades na automação da publicação e manutenção de dados abertos conectados usando o LodFlow

Bruno Eduardo D'Angelo de Oliveira

Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

bruno.eduardo.do@gmail.com

Abstract. *The concept of linked data arose so that the publication and dissemination of information on the web could be improved. However this process is still very costly for the user. To try to mitigate the difficulty found in the process the tool SysLODFlow was created, which integrates several solutions of the area creating a workflow that encompasses all stages of the publishing process. However, the software has limitations in its functionalities that detract from the user experience. The objective of this work is to review the tool's design, analyze the outstanding needs, and create the necessary solutions to the problems investigated.*

Resumo. *O conceito de dados conectados surgiu para que a publicação e difusão de informações na web pudessem ser aprimoradas. Contudo esse processo ainda é muito oneroso para o usuário. Para tentar abrandar a dificuldade encontrada no processo foi criada a ferramenta SysLODFlow, a qual integra diversas soluções da área criando um fluxo de trabalho que engloba todas as etapas do processo de publicação. Entretanto o software desenvolvido apresenta limitações em suas funcionalidades que prejudicam a experiência de utilização. O objetivo deste trabalho é rever a concepção da ferramenta, analisar as necessidades pendentes, e criar as devidas soluções aos problemas averiguados.*

1. Introdução

O conceito de *Linked Data* (dados conectados) é um conjunto de práticas introduzidas por Tim Berners-Lee (2006), que tem como objetivo utilizar a *web* para relacionar dados que previamente não tinham ligação, ou usar os recursos *web* para ao menos diminuir as barreiras entre as mesmas. Com isso em mente pode-se aplicar as técnicas descritas a grandes massas de dados, como, por exemplo, os disponibilizados pelo governo federal Brasileiro, assim possibilitando uma exploração mais fluida, e a possível descoberta, ou criação, de conhecimento a partir dos resultados obtidos.

No entanto, junto com todas as vantagens que o seguinte modelo proposto fornece, surgem as dificuldades em se realizar a manutenção dos mesmos, como evidenciado em

diversos estudos sobre o assunto de *Linked Data* [AUER et al., 2013]. Sendo essa uma atividade complicada e que envolve diversos passos que demandam ferramentas específicas. Porém o custo dessas operações pode vir a ser reduzido com a introdução da automatização do fluxo de trabalho e publicação. Para se atingir esse objetivo foi desenvolvido o *software* LODFlow [RAUTENBERG et al. 2015], o qual proporciona métodos para suportar o ciclo de vida destes conjuntos de dados de uma forma sistemática. Mas mesmo utilizando essa solução a carga imposta aos usuários ainda era considerada elevada, pois o tempo necessário para compreensão de seu uso era alto, e ainda assim era necessário utilizar outras aplicações.

Com o intuito de automatizar ainda mais o fluxo de trabalho criou-se o *SysLODFlow*, que faz uso das funcionalidades do LODFlow, mas fornece opções mais "*user friendly"* (e automatizadas) para se realizar o ciclo de publicação e manutenção de dados conectados. A ferramenta também foi desenvolvida de maneira a ser um aplicativo web, o que soluciona algumas restrições iniciais do LODFlow, e abrem as portas para funcionalidades que antes não eram possíveis em aplicações *standalone*, mas que trazem consigo algumas necessidades pontuais de design para web.

Todavia existem limitações no escopo de operação da aplicação que precisam ser tratadas e desenvolvidas, para que o programa possa melhor suprir as necessidades de seu público alvo. Dessa forma este trabalho pretende estudar a implementação do SysLODFlow, entender suas escolhas de design, propor e desenvolver, sempre se utilizando de boas práticas de engenharia e programação de *software*, soluções para as deficiências encontradas na iteração atual do projeto, bem como elaborar um documento que explique o funcionamento básico das funcionalidades do projeto.

2. Linked Data

Introduzido por Tim Berners-Lee (2006) a disciplina de *Linked Data*, ou Dados Conectados, trata de ilustrar uma nova forma de se publicar conhecimento e dados na internet. O termo *Linked Data* refere-se ao conjunto de melhores práticas para publicação e conexão de dados estruturados na *internet* [HEATH, 2009], sendo importante notarmos que este conceito originou-se de outro publicado por Berners-Lee: a *Web* Semântica. O autor ainda vai mais além e descreve *Linked Data* como "a *Web* Semântica feita de maneira correta" [BERNERS-LEE, 2008].

Dessa forma diz-se que a *Web* de Dados difere da *web* comum no fato de que a segunda navega através de *links* em documentos, enquanto a primeira se utiliza das relações para navegar em uma rede de conceitos [BERNERS-LEE et al., 2006]. Algumas tecnologias sustentam essa ideia, para nomear uma parte, temos: URIs, HTTP e o formato de publicação RDF. Assim, enquanto a *web* convencional se utiliza de *hyperlinks*, *Linked Data* utiliza o padrão RDF para criar declarações sobre dados arbitrários, que sejam relacionados entre si.

Em um de seus documentos técnicos Berners-Lee (2006) estabeleceu quatro princípios para a publicação e conexão de dados sob sua arquitetura:

5. Utilize URIs como nomes para seus recursos;
6. Utilize URIs HTTP para que seja possível que pessoas visualizem os recursos;
7. Quando um indivíduo visualizar o recurso, provenha informações úteis, utilizando os formatos padrões (SPARQL e RDF);
8. Inclua *links* para outras URIs, para que seja possível descobrir mais coisas.

Os princípios acima fornecem uma forma padrão para se publicar dados conectados na *web* clássica, pois aderem aos esquemas que vigoram no ambiente virtual. Através das tecnologias, modelos e princípios, discutidos aqui, torna-se possível publicar informações na *internet* de uma maneira mais inteligente, e que contribua para a iniciativa de uma rede de mundial computadores mais eficiente.

2.1 Linked Open Data

O conceito de *linked open data* (também conhecido como dados abertos conectados ou LOD) é muito similar ao de *linked data* convencional, contudo este último nem sempre precisa ser de livre acesso, enquanto o último exige tal qualidade.

Tim Berners Lee (2006) descreve LOD como sendo dados conectados publicados sob uma licença aberta que não impeça sua reutilização livre de custo. Dessa maneira pode-se dizer que o campo de dados abertos conectados é onde *linked data* e dados abertos se encontram.

3. Linked Data Workflow Management System

Grande parte do trabalho associado ao processo de *Linked Data* vem das tarefas relacionadas a preparação das informações para transformação para o formato RDF. Normalmente o processamento envolvido nesta etapa se dá através de *scripts* de aplicações proprietárias ou por fluxos de execução, e trabalho, manuais. Assim a extração e preparo das informações se torna extremamente custosa, em razão dos diversos passos exigidos - os quais devem ser feitos de maneira manual pelo operador dos instrumentos de software. Algumas dessas ações, não levando em conta a parte de disponibilização das informações, são: carga de dados brutos, extração e análise.

Como já discutido previamente, o *Linked Data Workflow Management System* (LDWMS), também conhecido de maneira abreviada como *LODFlow*, é um programa que visa auxiliar no processo de preparação e publicação desse tipo de dado, conseguindo esse feito ao prover um ambiente propício para o planejando, execução, reutilização e documentação de *workflows* de dados conectados. Na figura 1 pode-se observar o ciclo de vida que é apoiado na implementação deste programa.

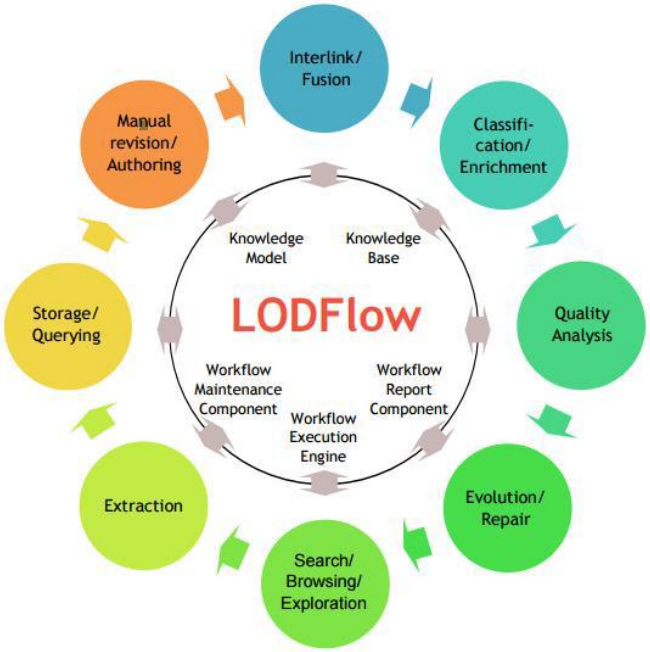


Figura 1. Ciclo de Vida suportado pelo LODFlow [RAUTENBERG et al. 2015].

Para conseguir sustentar completamente o ciclo ilustrado, soluções para modelagem, documentação, registro, manutenção e criação de relatórios, são disponibilizadas. Dessa maneira é possível reunir a maior parte das necessidades básicas, dos usuários que buscam publicar *Linked Data*, em uma única aplicação.

4. SysLODFlow

Como já discutido anteriormente, todo o processo de publicação e manutenção de dados conectados exige uma miríade de conhecimentos relacionados ao processo de *Linked Data*, bem como acerca de ferramentas de *software*, para que ao final se tenha um *dataset* de qualidade e que seja utilizável para *upload* à *internet*. O advento do LODFlow trouxe a possibilidade de se suportar todo o ciclo de vida de conjuntos de *Linked Data*, em um único artefato de *software*. Para isso são disponibilizadas componentes específicos que dão suporte aos diferentes estágios do processo, assim permitindo que a ferramenta seja utilizada na criação de conjuntos ou manutenção dos mesmos.

Contudo, a carga ao usuário ainda é elevada, visto que o LODFlow demanda tempo considerável para que seja dominado, exigindo tanto conhecimentos específicos quanto trabalho para processamento das informações. Motivados pela alto esforço e conhecimento, Morais e Morais (2016) decidiram criar uma aplicação *web*, batizada de SysLODFlow, que facilitasse o processo como um todo. Em suas palavras eles desenvolveram uma "ferramenta gráfica customizada para registro e manipulação das instâncias das classes envolvidas no LODFlow em sua base de conhecimento, a ontologia LDWPO. O objetivo do SysLODFlow acabou por ser: criar um ambiente customizado para registros de projeto, *workflows*, passos e execuções, para, deste modo, melhorar a experiência do usuário final ao facilitar o

entendimento dos processos envolvidos com a manipulação de ontologias, através do encapsulamento de todos os conceitos necessários para operação da LDWPO na aplicação desenvolvida.

Os aspectos técnicos da aplicação envolvem o fato de que a mesma deve rodar em um servidor Linux e que a linguagem de programação escolhida para sua criação sendo o Java. O *software* também foi desenvolvido seguindo uma estrutura em 3 camadas, as quais podemos chamar de: cliente, aplicação e fonte de dados [LIU; HEO; SHA, 2005], além da nomenclatura adotada originalmente. Abaixo o *stack* de tecnologias empregadas pode ser observado na figura 2.

Figura 2. Esquema representando as camadas de aplicação e seus componentes [Morais e Morais 2016].

O funcionamento da aplicação se dá através da criação de projetos de publicação de *Linked Data*, os quais contêm *workflows* que podem ser executados quando necessário. Os *workflows* em si são compostos por operações chamadas *steps*, que formam o fluxo de execuções que compõem o processo. A figura 3 nos mostra o diagrama de casos de uso do SysLODFlow, dando uma idéia melhor de todas as operações que podem ser realizadas.

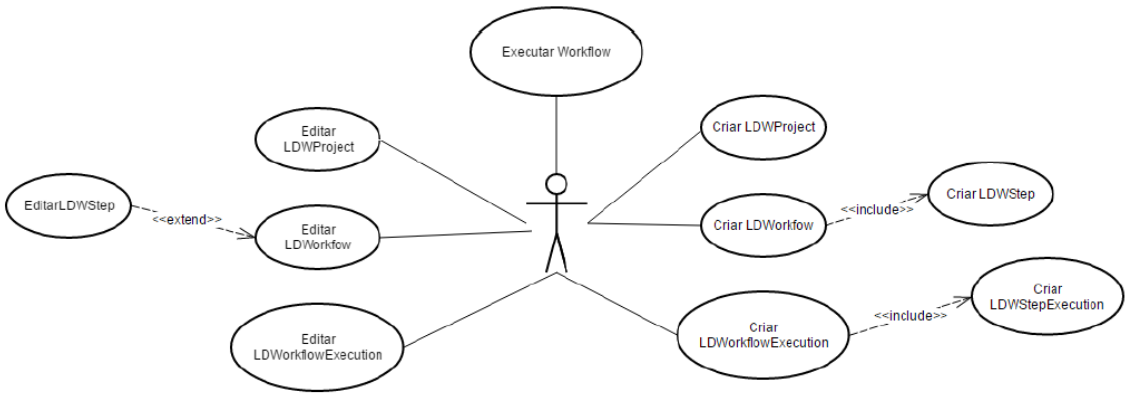
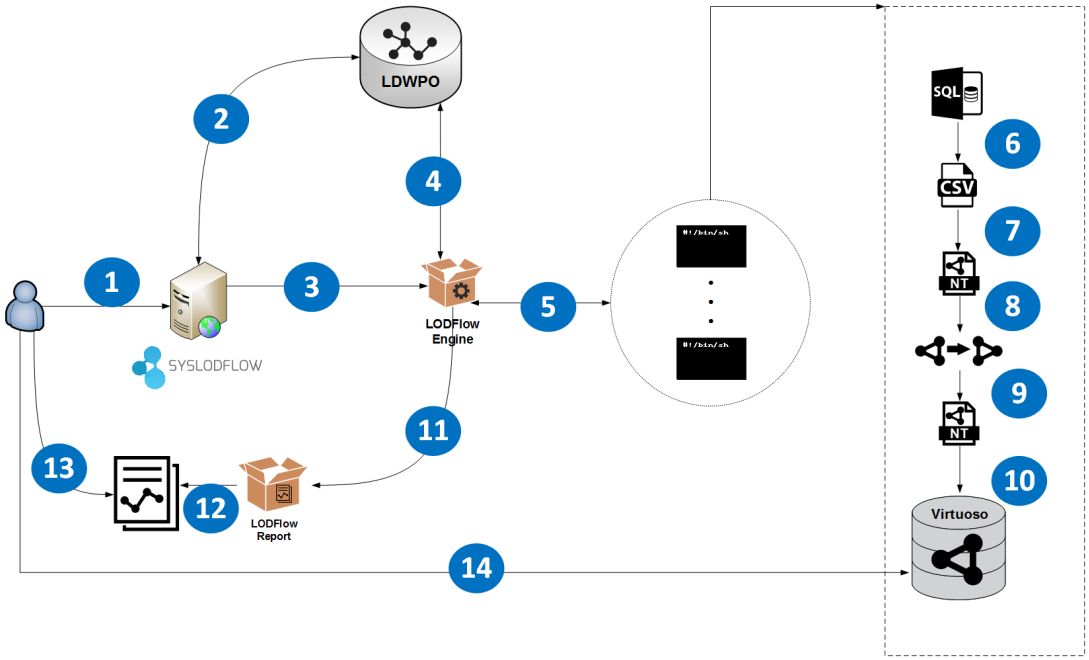


Figura 3. Casos de uso do SysLODFlow [Morais e Morais 2016].

Na figura 4 podemos ver os casos de uso descritos em ação formando o processo de publicação suportado pelo aplicativo, também podemos observar as interações do SysLODFlow com as ferramentas de apoio.



1. Usuário acessa SYSLODFLOW através de seu navegador, realiza as operações devidas e mandar executar o fluxo.
2. SYSLODFLOW armazena dados na ontologia
3. SYSLODFLOW chama a execução do LODFlow Engine para execução do fluxo informando o fluxo a ser executado
4. LODFlow Engine recupera da ontologia os dados relativos ao workflow a ser executado
5. LODFlow Engine realiza chamadas via script para execução de ferramentas de apoio, configuradas nos passos
6. Dados são extraídos de um banco de dados relacional (MySQL) e convertidos para CSV
7. Sparqlify realiza o mapeamento dos dados contidos no CSV e converte para triplas em formato NT
8. Arquivos de triplas é armazenado no banco de dados Virtuoso
9. Triplas são recuperadas do virtuoso e através do LIMES ligadas ao DBPedia
10. Triplas já ligadas ao DBPedia são armazenadas novamente no triple store Virtuoso
11. LODFlow engine aciona LODFlow Report para geração de relatório de execução do workflow
12.LODFlow Report gera relatório em HTML contendo informações acerca da execução
13. Usuário pode realizar o acesso ao relatório em HTML para verificar status da execução
14. Usuário pode realizar consultas sobre o banco de dados de triplas Virtuoso através de seu endpoint SPARQL

Figura 4. Ciclo de operação desenvolvido para o SysLODFlow [Morais e Morais 2016].

É importante destacar os pontos de que ao final da execução de um *workflow* dois artefatos são gerados: um relatório contendo os resultados da execução e um conjunto de dados RDF, bem como o fato de que todos componentes descritos podem ser salvos em uma base de dados, para que sejam recuperados e executados (ou alterados) em um momento posterior, de maneira a oferecer dinamismo ao procedimento de publicação e favorecendo o reuso de *workflows* de execução.

5. Avaliação do Sistema

Antes de se iniciar o processo de desenvolvimento das expansões das funcionalidades do SysLODFlow, era primeiramente necessário entender os seguintes fatores:

- O que a ferramenta almeja realizar;
- Como foram implementadas as funcionalidades;
- Como realizar as melhorias desejadas.

Para sanar a dúvida do primeiro ponto se recorreu ao documento de TCC que deu origem a ferramenta [MORAIS; MORAIS, 2016]. No documento o processo de elaboração e amparo teórico é explicado pelos autores, o que acabou por dar uma boa base para entendimento do que se queria alcançar. Durante esta etapa, compreendeu-se o desejo dos autores de se utilizar da ontologia LDWPO, para automatizar o processo de publicação de dados abertos conectados, como demonstrado por Rautenberg et al. (2016) em seu trabalho.

Além disso, o documento explica diversas decisões de design da ferramenta, bem como o funcionamento de certos aspectos da mesma. Contudo o diagrama de casos é o único tipo de arquivo voltado exclusivamente para documentação da ferramenta, o que faz com que os documentos que elaborem sobre os aspectos funcionais e teóricos do projeto sejam limitados ao relatório de TCC e artigos citados. Apesar de ambos serem boa fonte de informação, o objetivo desses não é dar um entendimento completo sobre a implementação dos recursos de *software* selecionados para o desenvolvimento, ou sobre a estruturação do código fonte do programa. Dessa maneira, fica aparente que uma documentação voltada para desenvolvedores não está presente nos arquivos do projeto. Ademais, ficou nítida a necessidade da realização da engenharia reversa das funcionalidades, a fim de se resgatar o design dos casos de uso já descritos na figura 3 e compreender as operações de uso geral.

No entanto, o primeiro desafio foi encontrado ao se tentar instalar corretamente o SysLODFlow. Erros provenientes do servidor de aplicação acusavam a falta de certas configurações, desconhecidas até o momento. Para resolver o impasse encontrado foi necessário entrar em contato com os autores do programa, somente dessa maneira foi possível compreender o que faltava ser configurado e colocar o SysLODFlow no ar.

De posse dos recursos citados, com a ferramenta instalada e funcionando, finalmente foi possível começar a compreender e traçar o funcionamento interno do SysLODFlow, através da exploração da interface gráfica, observação da execução do código fonte e das informações obtidas através de seu *debug*.

Durante os primeiros contatos com a ferramenta, tentou-se realizar os casos de uso previstos através da interface *web*. Percebeu-se durante esses procedimentos que muitas funcionalidades esperadas aparentavam não estar funcionando, não terem sido implementadas ou, apesar de implementadas, não estarem finalizadas. Muitas dessas suposições foram confirmadas ao se verificar as mensagens apresentadas pelo console de execução da IDE, bem como diretamente pelo código fonte, como pode ser observado na figura abaixo, a qual representa um botão sem funcionalidade atrelada.

```
<h:form id="form-download-owl">
    <h:commandButton id="downloadOwl"
    action="#" styleClass="btn btn-success btn-xs"
    value="#{msg['crud.file.download']}" />
</h:form>
```

Figura 5: Código do botão de download com propriedade de ação não implementada.

Além do mais, certas operações e configurações pareciam terem sido implementadas de maneira *hard coded*, o que vai de encontro com o propósito original da concepção do SysLODFlow.

A descoberta mais grave relacionada a esse tópico, se deu quando, uma chamada de execução a um programa Java externo foi descoberta, com seu endereço de localização apontando para uma pasta da máquina de um dos criadores do sistema (ilustrado na figura 6). Tal fato fazia com que, o fluxo de execução não fosse passível de reprodução fora do ambiente de desenvolvimento original, significando que parte essencial do aplicativo - a execução de *workflows* - não funcionava.

```
public void doExecute() {
    try {
        Runtime.getRuntime().exec("java -jar C:\\Users\\Jhonatan\\Downloads\\BancoDeClubes\\"
                                  + "BancoDeClubes\\BancoDeClubes\\dist\\BancoDeClubes.jar");
    } catch (IOException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

Figura 6: Função com endereço de arquivo hardcoded para máquina do autor original.

Diversos erros mais foram encontrados durante a etapa de análise do código fonte, fazendo com que ficasse claro o estado de imaturidade do projeto como um todo e de como a premissa de uma ferramenta funcional estava incorreta. Assim sendo, percebeu-se a necessidade de implementar e corrigir, as funcionalidades básicas que se supunha estarem operantes.

5.1. Erros Encontrados

São considerados erros os problemas que impeçam a utilização pretendida da ferramenta, gerem exceções no código e/ou vão contra regras e padrões de desenvolvimento para o desenvolvimento de aplicações *web*. Abaixo na tabela 1 pode-se ver todos os problemas encontrados durante a avaliação da ferramenta.

Tabela 1. Erros encontrados no sistema.

Erro Encontrado	Explicação	Solução Proposta
Layout geral das páginas do sistema.	Diversos componentes visuais, como ícones e botões, estão mal alinhados, ou excedendo a largura de seus contêineres. Múltiplas páginas seguem padrões diferentes de design, acarretando na não existência de regras para a composição das interfaces.	Aualizar os frameworks Bootstrap da versão 3 para 4 e SB Admin. Reestruturar o layout das páginas para que sigam uma lógica padrão e boas práticas de UX.
Botões apresentando diversos comportamentos errôneos ou não funcionando.	Botões “Salvar” e “Voltar”, de certas páginas, estão realizando verificações em campos fora de seu escopo; Botões para exclusão de arquivos não foram	Corrigir todos os erros relacionados de maneira individualizada, com as melhores soluções para cada caso.

	implementados; Botões para *download* de arquivos não foram implementados.	
Processo de criação e execução de LDWProjects e arquivos de ontologia.	Processo de criação de arquivos de ontologia implementado de maneira incorreta e inacabada.	Corrigir e finalizar o processo de criação dos arquivos de ontologia LDWPO, para que seja possível executar os *workflows*.
Arquivos XHTML com erros de sintaxe.	Arquivos XHTML com erros em sua sintaxe e com problemas de identação.	Corrigir os erros de sintaxe e identar de maneira correta os arquivos.
Strings com erros.	Strings apresentando erros de escrita.	Corrigir os erros para que o texto seja consistente e correto.
Exceções não tratadas.	Exceções de execução não tendo tratamento ideal.	Adicionar tratamento para as exceções observadas.
Classe usuário com propriedades conflitantes.	Classe usuário com propriedades conflitantes com a tabela criada no banco de dados relacional.	Excluir as propriedades extras da classe, ou mudar a tabela relacional para acomodar tais campos.
Projetos novos não separados em pastas distintas.	Os arquivos de ontologia são todos salvos sob o mesmo diretório.	Fazer com que cada arquivo de ontologia seja criado sob diretórios condizentes com o projeto e usuário relacionados.
Documentos de projeto sendo criados mesmo que o processo não tenha sido completado corretamente.	Caso o processo de criação de projetos seja abortado, ou um erro seja lançado, os arquivos de ontologia terão sido instanciados e salvos em disco de maneira corrompida.	Fazer com que os arquivos de ontologia só sejam salvos em disco após o processo ter sido finalizado corretamente.

5.2. Melhorias Encontradas

São consideradas melhorias: novas adições as funcionalidades já implementadas, ou a elaboração de novos componentes e/ou recursos, que expandam ou melhorem a operação da ferramenta. A tabela 2 ilustra quais áreas do sistema podem ser melhoradas.

Tabela 2. Melhorias propostas.

Melhoria Encontrada	Justificativa	Solução Proposta
Documentação básica.	A falta de arquivos de documentação cria uma barreira para os novos operadores da ferramenta, que acabam por ter de descobrir de maneira	Criar um documento de documentação README no formato *markdown.*

	exploratória como realizar as tarefas que desejam.	
Incluir padrão de entrada JSON.	O formato é leve e de fácil leitura e escrita, além de ser amplamente utilizado nos dias de hoje.	Adicionar suporte para arquivos de entrada de dados no formato JSON.
Adicionar recursos para cancelar operações.	Múltiplas seções da interface não apresentam um modo de cancelar o procedimento sendo realizado, ou de retornar a página que levou o operador até aquela região do *software*.	Adicionar algo como botões nas páginas relevantes, para que o fluxo de execução, bem como a experiência de uso, sejam aprimorados.
Criar testes automatizados.	Testes automatizados auxiliam no desenvolvimento de novas *features*, bem como ajudam a desenvolver código mais funcional e correto.	Criar testes unitários e de aceitação.
Interfaces Para Geração ou Edição de Arquivos de Configuração.	Os arquivos de configuração do LIMES e Sparqlify precisam ser criados préviamente e carregados para o projeto. Interfaces de criação desses arquivos facilitaria o processo.	Criar telas especializadas em criar os arquivos no formato XML e SML necessários para a operação customizada do LIMES e Sparqlify.

5.3. Soluções e Melhorias Implementadas

Devido ao tempo limitado, a necessidade de se realizar engenharia reversa no código do *software* e todas as funcionalidades não implementadas corretamente, nem todas as soluções propostas puderam ser realizadas. O que se fez foi priorizar as que teriam o maior impacto no sistema e que eram fundamentais para que o mesmo passa-se a atender os requisitos funcionais iniciais [Morais e Morais 2016].

Abaixo a tabela 3 descreve quais soluções foram implementadas, bem como o que foi feito, pois algumas propostas iniciais não foram seguidas a risca ou substituídas por inteiro.

Tabela 3. Soluções implementadas.

Erro/Melhoria	Implementação
Layout geral das páginas do sistema.	Frameworks atualizados, um mockup de tela foi criado como guia para novas páginas, navegação por *breadcrumbs* foi adicionada.
Adicionar recursos para cancelar operações.	Com a adição da navegação por *breadcrumbs* este problema foi resolvido, não necessitando de trabalho extra.
Botões apresentando diversos comportamentos errôneos ou não	Todas as funcionalidades faltantes foram implementadas;

funcionando.	Remoção de botões “Voltar” em prol da navegação por *breadcrumbs*; Correção dos bugs relacionados aos botões.
Documentação básica.	Criado arquivo README abrangendos os principais tópicos de interesse, bem como pontos onde se teve dificuldades durante o trabalho.
Classe usuário com propriedades conflitantes.	Removidas as prorpiedades conflitantes com a tabela do banco de dados.
Documentos de projeto sendo criados mesmo que o processo não tenha sido completado corretamente.	Recomendações iniciais adotadas.
Processo de criação e execução de LDWProjects e arquivos de ontologia.	Recomendações inciais adotadas, mas parcialmente implementadas devido a constrições de tempo.
Correção de arquivos XHTML com erros de sintaxe.	Recomendações iniciais adotadas.

6. Conclusão

Chegando ao final dos trabalhos realizados foram geradas centenas de novas linhas de código, diversas dependências atualizadas e dezenas de *commits* submetidos ao repositório GitHub que hospeda o projeto, que acabaram por fazer com que o SysLODFlow se tornasse uma ferramenta muito mais completa, funcional e que atende aos requisitos funcionais estabelecidos e descritos em sua concepção inicial. A tabela 4 fornece a visão geral sobre quais propostas elencadas foram adotadas, parcialmente implementadas ou não implementadas.

Tabela 4. Visão geral da situação dos pontos elencados.

Erro e/ou Melhoria Relacionada	Situação das Soluções
Layout geral das páginas do sistema.	Implementadas.
Botões apresentando diversos comportamentos errôneos ou não funcionando.	Implementadas.
Processo de criação e execução de LDWProjects e arquivos de ontologia.	Parcialmente Implementadas.
Arquivos XHTML com erros de sintaxe.	Implementadas.
Strings com erros.	Implementadas.
Exceções não tratadas.	Implementadas.
Classe usuário com propriedades conflitantes.	Implementadas.

Projetos novos não separados em pastas distintas.	Não implementadas.
Documentos de projeto sendo criados mesmo que o processo não tenha sido completado corretamente.	Implementadas.
Documentação básica.	Implementadas.
Incluir padrão de entrada JSON.	Não implementadas.
Adicionar recursos para cancelar operações.	Solução alternativa implementada.
Criar testes automatizados.	Não implementadas.
Interfaces Para Geração ou Edição de Arquivos de Configuração.	Não implementadas.

Todas as alterações, e código fonte do aplicativo, podem ser encontradas no endereço *github.com/bruno-edo/syslodflow_01*, onde o projeto estará perpétuamente disponível. Ademais, seguindo o espírito do *software* livre, novos colaboradores e contribuições serão prontamente aceitas após revisão das alterações submetidas, para a certificação de que estão de acordo com as regras de negócio estabelecidas.

7. Trabalhos Futuros

De posse do conhecimento gerado neste TCC as possibilidades de desenvolvimento são amplas, mas algumas delas acabam por ser mais urgentes que outras - já que não se pode tocar em todos os pontos que se desejava durante a etapa de codificação deste trabalho. Com isso em mente é recomendado que no futuro sejam desenvolvidos os seguintes tópicos:

- Criação de testes automatizados que aumentem a confiabilidade de novas submissões (*commits*) ao código do SysLODFLow, bem como auxilie os desenvolvedores;
- Elaboração de interface para criação e edição de arquivos de configuração para as ferramentas de apoio Sparqlify e LIMES;
- Criação de interface para a reutilização de propriedades de projetos antigos no cadastro de novos projetos e *workflows*;
- Adição de suporte a novas ferramentas de apoio que realizem tarefas semelhantes ao LIMES e Sparqlify;
- Expansão e melhoria da GUI do *app*, seguindo os conceitos de *User Experience*, boas práticas, usabilidade e acessibilidade;
- Expansão e melhoria do suporte a múltiplos usuários cadastrados e utilizando a aplicação, para que se passe a ter um ambiente com diversos usuários criando e compartilhando projetos próprios (com a devida separação de arquivos e tratamento de exceções);
- Documentação do projeto mais aprofundada, abrangente e técnica. Cobrindo aspectos como funcionamento dos métodos e ferramentas de apoio;

- Tratamento de quaisquer erros, exceções ou melhorias discutidos anteriormente neste trabalho, mas que não puderam ser desenvolvidos devido a constrições de tempo;
- Desenvolvimento novos modelos de *workflows* que podem ser adicionados a projetos criados;
- Melhoria e expansão do SysLODFlow no que tange a aspectos fora do escopo inicial da aplicação.

Referências

BERNERS-LEE, Tim. Linked Data: design issues. 2006. Disponível em: <https://www.w3.org/DesignIssues/LinkedData.html>. Acesso em: 07 nov. 2016.

AUER, Sören et al. Introduction to Linked Data and Its Lifecycle on the Web. Reasoning Web. Semantic Technologies For Intelligent Data Access, [s.l.], p.1-90, 2013. Springer Science + Business Media. http://dx.doi.org/10.1007/978-3-642-39784-4_1.

RAUTENBERG, Sandro et al. LODFlow: a workflow management system for linked data processing. Proceedings Of The 11th International Conference On Semantic Systems - Semantics '15, [s.l.], p.137-144, set. 2015. Association for Computing Machinery (ACM). http://dx.doi.org/10.1145/2814864.2814882. Disponível em: <http://eis.iai.uni-bonn.de/blog/wp-content/uploads/2015/08/SEMANTiCS_2015_Research_Track_submission_96.pdf>. Acesso em: 2 nov. 2016.

HEATH, Tom. Linked Data Organization. Linked Data - Connect Distributed Data across the Web: Frequently Asked Questions (FAQs). 2009. Disponível em: <http://linkeddata.org/faq>. Acesso em: 05 jun. 2017.

BERNERS-LEE, Tim. The Linked Open Data Movement. [s.i]: W3c, 2008. 22 slides, color. Disponível em: <https://www.w3.org/2008/Talks/0617-lod-tbl>. Acesso em: 05 jun. 2017.

BERNERS-LEE, Tim et al. Tabulator: Exploring and Analyzing linked data on the Semantic Web. Proceedings Of The 3rd International Semantic Web User Interaction Workshop: SWUI, Massachussets, v. 1, n. 1, p.1-16, jan. 2006. Disponível em: <http://hmslydia.com/my_publications/BernersLeeTabulator2006.pdf>. Acesso em: 05 jun. 2017.

MORAIS, Jean Carlos de; MORAIS, Jhonatan Carlos de. SysLodFlow: Uma Ferramenta de Apoio a Automação Da Publicação e Manutenção de Linked Data Utilizando o LODFlow. 2016. 77 f. TCC (Graduação) - Curso de Sistemas de Informação, Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis, 2016.

LIU, Xue; HEO, Jin; SHA, Lui. Modeling 3-Tiered Web Applications. 13th Ieee International Symposium On Modeling, Analysis, And Simulation Of Computer And Telecommunication Systems, [s.l.], v. 1, n. 1, p.1-8, set. 2005. IEEE. http://dx.doi.org/10.1109/mascots.2005.40.